Branching within Time: an Expressively Complete and Elementarily Decidable Temporal Logic for Time Granularity

MASSIMO FRANCESCHET

Dipartimento di Matematica e Informatica, Università di Udine, Via delle Scienze, 206 – 33100 Udine, Italy.
E-mail: francesc@dimi.uniud.it

ANGELO MONTANARI

Dipartimento di Matematica e Informatica, Università di Udine, Via delle Scienze, 206 – 33100 Udine, Italy.
E-mail: montana@dimi.uniud.it

ABSTRACT:

Suitable extensions of monadic second-order theories of k successors have been proposed in the literature to specify in a concise way reactive systems whose behaviour can be naturally modeled with respect to a (possibly infinite) set of differently-grained temporal domains. This is the case, for instance, of the wide-ranging class of real-time reactive systems whose components have dynamic behaviours regulated by very different time constants, e.g., days, hours, and seconds. In this paper, we focus on the theory of k-refinable downward unbounded layered structures $\mathrm{MSO}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$*, that is, the theory of infinitely refinable structures consisting of a coarsest domain and an infinite number of finer and finer domains, whose satisfiability problem is* nonelementarily *decidable. We define a propositional temporal logic counterpart of* $\mathrm{MSO}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$ *with set quantification restricted to infinite paths, called* $\mathrm{CTSL}_{\mathrm{k}}^{*}$*, which features an original mix of linear and branching temporal operators. We prove the expressive completeness of* $\mathrm{CTSL}_{\mathrm{k}}^{*}$ *with respect to such a* path *fragment of* $\mathrm{MSO}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$ *and show that its satisfiability problem is 2EXPTIME-complete.*

KEYWORDS: *temporal logics, time granularity, specification and verification of programs*

1 Introduction

The ability of providing and relating temporal representations at different 'grain levels' of the same reality is widely recognized as an important research theme for temporal logic and a major requirement for many applications, including formal specifications, artificial intelligence, temporal databases, and data mining, e.g. [2, 6, 11].

A systematic framework for *time granularity*, based on a many-level view of temporal structures, with matching logics and decidability results, has been proposed in [21]. The many-level temporal structure replaces the flat temporal structure of standard temporal logics by a temporal universe consisting of a (possibly infinite) set of related differently-grained temporal domains. Such a temporal universe identifies the relevant temporal domains and defines the relations between time points belonging to different domains. Suitable temporal operators make it possible to specify the temporal domain(s) a given formula refers to (*contextualization*), as well as to constrain the relationships between formulae within any given domain (*local displacement*) and across temporal domains (*projection*). The language for time granularity is the second-order language $\mathrm{MSO}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$, where $<_{tot}$ is a total ordering over the temporal universe and, for every element x of the temporal universe, $\downarrow_0 (x), \ldots, \downarrow_{k-1} (x)$ are the k elements of the immediately finer temporal domain (if any) into which x is refined. Such a language can be interpreted over both n-layered and ω-layered structures. The corresponding theories have been proved to be expressive enough to capture the key features of metric and layered temporal logics, that is, to define contextualization, displacement, and projection operators [21].

In [26], the decidability of the theory of k-refinable *n-layered structures* has been proved by mapping its decidability problem into an equivalent one relative to the finest layer, and, then, by reducing such a problem to the decidability problem for the monadic second-order theory of one successor $\mathrm{MSO}[<]$ [4]. The class of *k-refinable upward unbounded layered structures*, i.e., ω-layered structures consisting of a finest temporal domain and an infinite number of coarser and coarser domains (cf. Figure 1), and the class of *k-refinable downward unbounded layered structures*, i.e., infinitely refinable structures consisting of a coarsest domain and an infinite number of finer and finer domains (cf. Figure 2), have been investigated in [23]. Both theories have been shown to be *nonelementarily* decidable: the first one has been reduced to the monadic second-order theory of one successor, properly extended with a suitable partition function (the resulting theory has been proved to be the counterpart, in the style of Büchi Theorem, of the class of ω-languages accepted by k-ary tree systolic automata, which strictly includes the class of ω-regular languages); the second one has been reduced to the monadic second-order theory of k successors (with $k \geq 2$). An expressively complete temporal logic counterpart of $\mathrm{MFO}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$, the first-order fragment of the theory of k-refinable upward unbounded layered structures, has been proposed in [24]. In this paper, we address the problem of finding a temporal logic counterpart to a suitable fragment of the theory of k-refinable downward unbounded layered structures.

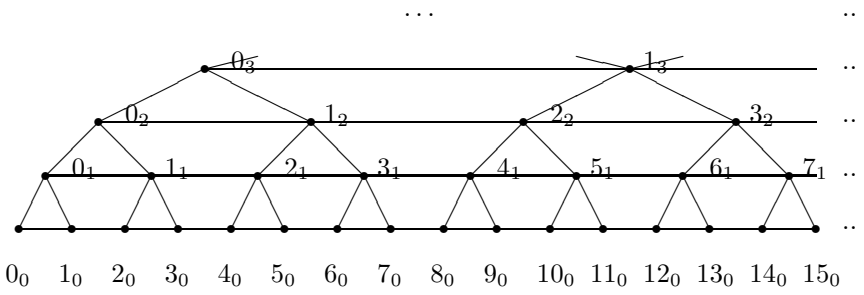

FIG. 1. A 2-refinable upward unbounded layered structure.

We first provide an alternative view of these structures as infinite sequences of infinite k-ary trees. Then, we define a temporal logic counterpart of the theory of k-refinable downward unbounded layered structures $\mathrm{MSO}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$ with set quantification restricted to infinite tree paths, $\mathrm{MPL}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$ for short, and show that it is expressively complete. The resulting logic, that we called $\mathrm{CTSL}_{\mathrm{k}}^{*}$ (Computational Tree Sequence Logic with k successors), can be viewed as a combined logic that embeds a logic for branching within (discrete) time points into a linear time logic. This form of logic combination is called *temporalization* in [13]. The nice feature of temporalization is that it transfers many logical properties, such as soundness and completeness of axiomatisations and decidability, from the component logics to the combined one.

In order to prove the expressive completeness of $\mathrm{CTSL}_{\mathrm{k}}^{*}$, we define a translation of $\mathrm{MPL}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$ formulas into $\mathrm{CTSL}_{\mathrm{k}}^{*}$ ones that is based on a model-theoretic decomposition lemma for tree sequences, which is proved by exploiting an application of the Ehrenfeucht game. Compared with $\mathrm{MPL}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$, $\mathrm{CTSL}_{\mathrm{k}}^{*}$ presents two major advantages. First, the satisfiability problem for $\mathrm{CTSL}_{\mathrm{k}}^{*}$ is elementarily decidable, while that for $\mathrm{MPL}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$ is nonelementarily decidable (we will prove that the satisfiability problem for $\mathrm{CTSL}_{\mathrm{k}}^{*}$ admits a decision procedure whose computational time complexity is doubly exponential in the length of the input formula, and that it is 2EXPTIME-hard). Furthermore, according to the combining logic perspective [12, 13, 14, 15], an axiomatisation (resp. model and satisfiability checking procedures) for $\mathrm{CTSL}_{\mathrm{k}}^{*}$ can be synthesized from axiomatisations (resp. model and satisfiability checking procedures) for the component logics.

The paper is organized as follows. In Section 2 we illustrate the rationale of the work. In Section 3 we introduce the relevant theories of downward unbounded layered structures. In Section 4 we define the logic $\mathrm{CTSL}_{\mathrm{k}}^{*}$. In Section 5 we study its expressiveness and complexity. In Section 6 we show how to tailor the logical framework for downward unbounded layered structures to cope with n-layered ones. Conclusions provide an assessment of the work done, and outline future research directions.

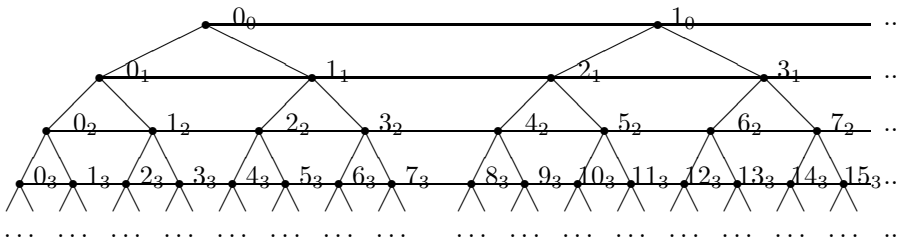

FIG. 2. A 2-refinable downward unbounded layered structure.

2 Rationale

The original motivation of our research was the design of a temporal logic embedding the notion of time granularity, suitable for the specification of complex (real-time) reactive systems whose components evolve according to different time units. However, there are significant similarities between the problems we encountered in pursuing our goal, and those addressed by current research on combining logics, theories, and structures. Furthermore, we recently established interesting connections between multi-level temporal logics and automata theory that suggests a complementary point of view on time granularity: besides an important feature of a representation language, time granularity can be viewed as a formal tool to investigate expressiveness and decidability properties of temporal theories. Finally, as a by-product of our work, we defined a uniform framework for time and states that reconciles the tense logic and the logic of programs perspectives.

2.1 The specification of granular real-time reactive systems

Timing properties play a major role in the specification of reactive and concurrent software systems that operate in real-time, which are among the most critical software systems. They constrain the interactions between different components of the system as well as between the system and its environment, and minor changes in the precise timing of interactions may lead to radically different behaviors. Temporal logic has been successfully used for modeling and analyzing the behavior of reactive and concurrent systems, e.g. [20]. It supports semantic model checking, which can be used to verify the consistency of specifications, and to check positive and negative examples of system behavior against specifications; it also supports pure syntactic deduction, which may be used to prove properties of systems. Unfortunately, most common specification languages are inadequate for real-time applications, because they lack an explicit and quantitative representation of time. A few remarkable exceptions do exist. They are extensions of Petri Nets or metric variants of temporal logic, which support direct and quantitative specifications of temporal properties and relevant vali-

dation activities.

There are, however, systems whose temporal specification is far from being simple even with timed Petri Nets or metric temporal logic. Consider the wide-ranging class of real-time systems whose components have dynamic behaviours regulated by very different—even by orders of magnitude—time constants (hereafter *granular real-time reactive systems*). As an example, consider a pondage power station consisting of a reservoir, with filling and emptying times of days or weeks, generator units, possibly changing state in a few seconds, and electronic control devices, evolving in milliseconds or even less. A complete specification of the power station must include the description of these components and of their interactions. A natural description of the temporal evolution of the reservoir state will probably use days: "During rainy weeks, the level of the reservoir increases 1 meter a day", while the description of the control devices behaviour may use microseconds: "When an alarm comes from the level sensors, send an acknowledge signal in 50 microseconds". We say that systems of such a type have *different time granularities*. It is somewhat unnatural, and sometimes impossible, to compel the specifier to use a unique time granularity, microseconds in the previous example, to describe the behaviour of all the components. A good language must allow the specifier to easily describe all simple and intuitively clear facts (naturalness of the notation). Hence, a specification language for granular real-time reactive systems must support *different time granularities* to allow one (i) to maintain the specifications of the dynamics of differently-grained components as separate as possible (modular specifications), (ii) to differentiate the refinement degree of the specifications of different system components (flexible specifications), and (iii) to write complex specifications in an incremental way by refining higher-level predicates associated with a given time granularity in terms of more detailed ones at a finer granularity (incremental specifications).

2.2 The combining logic perspective

Even though the original motivation of our work on time granularity was the design of a temporal logic suitable for the specification of granular real-time reactive systems, there are significant similarities between the problems it addresses and those dealt with by the current research on logics that model changing contexts and perspectives. Indeed, even if it has been developed in a temporal framework, our proposal actually outlines the basic features of a general *logic of granularity*. In this respect, it can be seen as a generalization of the well-known Rescher and Garson's topological logic to layered structures [27]. Moreover, it presents interesting connections with the logics of contexts developed in the area of knowledge representation, where modalities are used to shift variables, domains, and interpretation functions from one context to another [5]. More generally, the design of these types of logics is emerging as a relevant research topic in the broader area of combination of logics, theories, and structures, at the intersection of logic with artificial intelligence, computer science, and computa-

tional linguistics [3]. In our work, we devised suitable combination techniques both to define temporal logics for time granularity and to prove their logical properties, such as axiomatic completeness and decidability [22, 26]. Furthermore, we followed the combining logic approach to build a model checking framework for granular reactive systems and logics [15].

2.3 A complementary point of view on time granularity

A complementary point of view on time granularity arises from interesting connections that link multi-level temporal logics and automata theory: time granularity can be viewed not only as an important feature of a representation language, but also as a formal tool to investigate the definability of meaningful timing properties, such as density and exponential grow/decay, as well as the expressiveness and decidability of temporal theories [23, 25]. In this respect, the number of layers (single vs. multiple, finite vs. infinite) of the underlying temporal structure, as well as the nature of their interconnections, play a major role: certain timing properties can be expressed using a single layer; others using a finite number of layers; others only exploiting an infinite number of layers. Timing properties of granular reactive systems composed by a finite number of differently-grained temporal components can be specified by using n-layered metric temporal logics. Furthermore, if provided with a rich enough layered structure, n-layered metric temporal logics suffice to deal with conditions like "p holds at all even times of a given temporal domain" that cannot be expressed using flat propositional temporal logics. ω-layered metric temporal logics allow one to express relevant properties of infinite sequences of states over a single temporal domain that cannot be captured by using flat or n-layered temporal logics. For instance, k-refinable upward unbounded layered logics allow one to express conditions like "p holds at all time points k^i, for all natural numbers i, of a given temporal domain", while downward unbounded ones allow one to constrain a given property to hold true 'densely' over a given time interval [25].

2.4 Reconciling tense logics and logics of programs

As pointed out in [25], logic and computer science communities have traditionally followed a different approach to the problem of representing and reasoning about time and states. Research in philosophy, linguistics, and mathematical logic resulted in a family of (metric) tense logics that take *time* as a primitive notion and define (*timed*) *states* as sets of atomic propositions which are true at given time points. Recently, a few papers demonstrated the possibility of successfully exploiting metric (possibly layered) tense logics in computer science, e.g. [21, 22]. On the other hand, most research in computer science concentrated on the so–called temporal logics of programs, which have been successfully used to specify and verify reactive and concurrent systems, e.g. [10]. In order to deal with real-time systems, such logics have been provided

with a metric of time, e.g. [1]. The resulting temporal logics, called *real-time logics*, take *state* as a primitive notion, and define *time* as an attribute of states. More precisely, given an ordered set of states $\mathcal{S}$ and an ordered set of time points $\mathcal{T}$, real-time logics are characterized by a weakly monotonic function $\rho : \mathcal{S} \to \mathcal{T}$ that associates a time instant with each state. Real-time logics allow one to model pairs of states $s_i, s_j \in \mathcal{S}$ such that $s_i < s_j$ and $\rho(s_i) = \rho(s_j)$ (temporally indistinguishable states) or $\rho(s_{i+1}) > \rho(s_i) + 1$ (temporal gaps between states). Providing metric temporal logics with an infinite number of layers (ω-layered), where each time point belonging to a given layer can be decomposed into k time points of the immediately finer one (k-refinable), makes it possible to reconcile metric tense logics and real-time logics of programs. The basic idea is that any pair of distinct states, belonging to the same course of events, can always be temporally ordered, provided that we can refer to a sufficiently fine temporal domain. Furthermore, the ordering between pairs of states, which are temporally indistinguishable with respect to the considered domain, is actually induced by their temporal ordering with respect to a finer domain, with respect to which their are temporally distinguishable. Notice that a finite number of layers is not sufficient to capture timed state sequences: it is not possible to fix a priori any bound on the granularity that a domain must have to allow one to temporally order a given pair of states, and thus we need to have an infinite number of temporal domains at our disposal. In [25], Montanari et al. show how to embed the theory of timed state sequences, underlying real-time logics, into the theories of upward and downward unbounded metric and layered temporal structures. Such an embedding allows one to deal with temporal indistinguishability of states and temporal gaps between states. In the granularity setting, temporal indistinguishability and temporal gaps can indeed be interpreted as phenomena due to the fact that real-time logics lack the ability to express properties at the right (finer) level of granularity: distinct states, having the same associated time, can always be ordered at the right level of granularity; similarly, time gaps represent intervals in which a state cannot be specified at a finer level of granularity.

3 Structures and theories of time granularity

In this section, we formally define k-refinable downward unbounded layered structures and the corresponding monadic second-order theories.

3.1 Infinite sequences of infinite k-ary trees

According to the original definition given by Montanari et al. in [23], k-refinable downward unbounded layered structures are triplets $\langle \bigcup_{i\geq 0} T^i, \downarrow, <_{tot} \rangle$, where

- $\{T^i\}_{i\geq 0}$ are pairwise disjoint copies of $\mathbb{N}$;
- $\downarrow$: $\{0, \ldots, k-1\} \times \bigcup_{i\geq 0} T^i \to \bigcup_{i\geq 1} T^i$ is a bijection such that, for all $i \geq 0$,

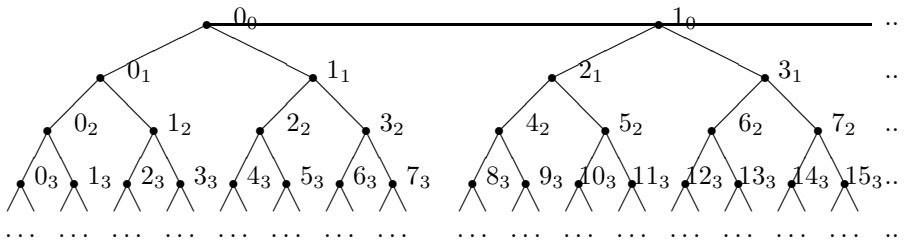

FIG. 3. A 2-refinable downward unbounded layered structure (revisited).

$\downarrow|_{T^i}\colon \{0,\ldots,k-1\} \times T^i \to T^{i+1}$ is a bijection;

- $<_{tot}$ is such that
  1. $\langle T^0, <_{tot}|_{T^0 \times T^0}\rangle$ is isomorphic to $\mathbb{N}$ with the usual ordering;
  2. $t <_{tot} \downarrow (j,t)$, for $0 \leq j \leq k-1$;
  3. $\downarrow (j,t) <_{tot} \downarrow (j+1,t)$, for $0 \leq j \leq k-2$;
  4. if $t <_{tot} t'$ and $t' \notin \bigcup_{i\geq 0} \downarrow^i (t)$, then $\downarrow (k-1,t) <_{tot} t'$;
  5. if $t <_{tot} t'$ and $t' <_{tot} t''$, then $t <_{tot} t''$.

  where $\downarrow^n (t) \subseteq \bigcup_{i\geq 0} T^i$ is the set such that $\downarrow^0 (t) = \{t\}$ and, for every $n \geq 1$, $\downarrow^n (t) = \{\downarrow (j,t') : t' \in \downarrow^{n-1} (t), 0 \leq j \leq k-1\}$.

It is worth noting that the domain $\bigcup_{i\geq 0} T^i$ is structured by $\downarrow$ in such a way that each time point of T_0 is the root of a leafless, perfectly balanced k-ary tree. The total ordering $<_{tot}$ over $\bigcup_{i\geq 0} T^i$ is induced by the linear ordering of the trees, for pairs of nodes belonging to different trees, and by the preorder visit of the tree, for pairs of nodes belonging to the same tree (cf. Figure 3). In the following, we make this remark more precise by providing an equivalent characterization of k-refinable downward unbounded layered structures as infinite sequences of infinite k-ary trees. Later, we will exploit this alternative formulation to obtain a temporal logic counterpart of the theory of k-refinable downward unbounded layered structures.

We start by defining the notion of Σ-labeled k-ary tree. Let $T_k = \{0,\ldots,k-1\}^*$ be the set of finite strings over $\{0,\ldots k-1\}$ and, for $x \in T_k$, let $|x|$ be the length of x. We denote by ϵ the empty string ($|\epsilon| = 0$). Let Σ be a fixed finite alphabet. An *infinite Σ-labeled k-ary tree*, with $k \geq 1$, is a map $t : T_k \to \Sigma$. For $k \geq 2$, $T_k(\Sigma)$ defines the set of infinite Σ-labeled k-ary trees, while, for $k = 1$, it defines the set of *infinite sequences* (or *infinite words*). A *finite sequence* (or *finite word*) is a map from an initial segment of T_1 to Σ. For any $0 \leq i \leq k-1$ and $x \in T_k$, let $\downarrow_i (x) = xi$ be the i-th *son* of x. Furthermore, we denote by $<$ the (partial) ordering induced by the *proper prefix* relation over T_k: $x < y$ iff there exists $z \neq \epsilon$ such that $xz = y$. The lexicographical (total) ordering on T_k, denoted by $<_{lex}$, is defined as follows: $x <_{lex} y$ iff $x < y$ or $x = zav$ and $y = zbu$, with $a, b \in \{0 \ldots k-1\}$ and $a < b$ (over $\mathbb{N}$). Any given tree

$t \in T_k(\Sigma)$ can be represented as a structure $\hat{t} = (T_k, <, (\downarrow_i)_{i=0}^{k-1}, (P_a)_{a\in\Sigma})$, where $P_a = \{x \in T_k \mid t(x) = a\}$, for every $a \in \Sigma$. Given $x \in T_k$, a *chain* in t, starting at x, is a subset of T_k, linearly ordered by $<$, such that x is the minimum of the chain. A *path* X in t, starting at x, is a chain, starting at x, such that there exists no chain Y, starting at x, which properly includes X. A *full path* in t is a path starting at the root ϵ of t, that is, a maximal, linearly ordered subset of t. The *i-th layer* of t is the finite set $\{x_0, \ldots, x_{k^i-1}\} \subset T_k$ such that, for $r = 0, \ldots, k^i - 1$, $|x_r| = i$.

An infinite sequence of infinite Σ-labeled k-ary trees is a map $ts : TS_k \to \Sigma$, where $TS_k = \mathbb{N} \times T_k$. The i-th tree of the sequence is the set $\{(i, x) \in TS_k \mid x \in T_k\}$. Let $TS_k(\Sigma)$ be the set of infinite sequences of Σ-labeled infinite k-ary trees. Given $0 \leq i \leq k-1$ and $(j, x) \in TS_k$, $\downarrow_i ((j, x)) = (j, xi)$ is the i-th *projection* of (j, x). A *total* ordering relation $<_{tot}$ over TS_k, which corresponds to the ordering relation $<_{tot}$ over $\bigcup_{i\geq 0} T^i$, can be defined as follows:

1. $(i, \epsilon) <_{tot} (j, \epsilon)$, $j > i \geq 0$;
2. $(i, x) <_{tot} \downarrow_j (i, x)$, for $0 \leq j \leq k-1$ and $i \geq 0$;
3. $\downarrow_j (i, x) <_{tot} \downarrow_{j+1} (i, x)$, for $0 \leq j \leq k-2$ and $i \geq 0$;
4. if $(i, x) <_{tot} (j, y)$ and either $i < j$ or it is not the case that $x < y$ (over T_k), then $\downarrow_{k-1} (i, x) <_{tot} (j, y)$, for all $i, j \geq 0$;
5. if $(i, x) <_{tot} (n, z)$ and $(n, z) <_{tot} (j, y)$, then $(i, x) <_{tot} (j, y)$.

Furthermore, the domain TS_k is *partially* ordered by two ordering relations $<_1$ and $<_2$ defined as follows: for every $(i, x), (j, y) \in TS_k$, $(i, x) <_1 (j, y)$ if and only if $i < j$ (over $\mathbb{N}$) and $(i, x) <_2 (j, y)$ if and only if $i = j$ and $x < y$ (over T_k). Any given tree sequence $ts \in TS_k(\Sigma)$ can be represented as a structure $\hat{ts} = (TS_k, <_{tot}, (\downarrow_i)_{i=0}^{k-1}, (P_a)_{a\in\Sigma})$, where $P_a = \{(j, x) \in TS_k \mid ts((j, x)) = a\}$, for every $a \in \Sigma$. Given $ts \in TS_k(\Sigma)$, $(j, x) \in TS_k$, and $0 \leq i \leq k-1$, we denote by $ts_{(j,x)}$ and $ts_{(j,x),i}$ the tree rooted at (j, x) and the tree rooted at the i-th son of (j, x), respectively. Accordingly, the *i-th tree* of ts is denoted by $ts_{(i,\epsilon)}$. A *chain* (resp. *path*, *full path*) in ts is a chain (resp. path, full path) in $ts_{(i,\epsilon)}$, for some $i \geq 0$. We denote by $\Pi(ts)$ the set of full paths of ts. The *i-th layer* L_i of ts is the set $\bigcup_{j\geq 0}\{(j, x_0), \ldots, (j, x_{k^i-1})\} \subset TS_k$ such that, for $r = 0, \ldots, k^i - 1$, $|x_r| = i$. A path P (resp. layer L) in ts can be ordered with respect to $<_{tot}$ obtaining a sequence $(j_1, x_1)(j_2, x_2)\ldots$; we denote by $P(i)$ (resp. $L(i)$) the i-th element (j_i, x_i) of such a sequence. It is worth noting that the restriction of $<_1$ to the elements belonging to the 0-layer is a total ordering, while this is not the case for all the other finer layers. We will take advantage of this fact when we will define a combined temporal logic for time granularity.

It is not difficult to show that any given Σ-labeled k-refinable downward unbounded layered structure $\langle \bigcup_{i\geq 0} T^i, \downarrow, <_{tot}\rangle$ corresponds to an infinite sequence of infinite Σ-labeled k-ary trees $\hat{ts} = (TS_k, <_{tot}, (\downarrow_i)_{i=0}^{k-1}, (P_a)_{a\in\Sigma})$, and vice versa.

3.2 Theories of downward unbounded layered structures

In this section, we define the relevant theories of k-refinable downward unbounded layered structures.

DEFINITION 3.1
Given a finite alphabet Σ, let $\mathrm{MSO}_\Sigma[c_1, \ldots, c_r, u_1, \ldots, u_s, b_1, \ldots, b_t]$ (abbreviated $\mathrm{MSO}_\Sigma[\tau]$) be the second-order language with equality over a set $C = \{c_1, \ldots, c_r\}$ of constant symbols, a set $U = \{u_1, \ldots, u_s\}$ of unary relational symbols, and a set $B = \{b_1, \ldots, b_t\}$ of binary relational symbols, which is defined as follows:

(i) *atomic formulas* are of the forms $x = y$, $x = c_i$, with $1 \leq i \leq r$, $u_i(x)$, with $1 \leq i \leq s$, $b_i(x, y)$, with $1 \leq i \leq t$, $x \in X$, and $x \in P_a$, where x, y are individual variables, X is a set variable, and P_a, with $a \in \Sigma$, is a monadic predicate;

(ii) *formulas* are built up from atomic formulas by means of the Boolean connectives $\neg$, $\wedge$, $\vee$, and $\rightarrow$, and the quantifiers $\forall$ and $\exists$, ranging over both individual and set variables.

$\mathrm{MSO}_\Sigma[\tau]$-formulas are interpreted over relational structures consisting of a domain $\mathcal{D}$ and an interpretation function $\mathcal{I}$ for the symbols in the vocabulary τ and the monadic predicates $(P_a)_{a\in\Sigma}$. Semantics structures for $MSO_\Sigma[\tau]$ give the same interpretation to symbols in τ; they may only differ in the interpretation of the predicates $(P_a)_{a\in\Sigma}$.

For any given vocabulary τ, let $\mathrm{MFO}_\Sigma[\tau]$ be the first-order fragment of $\mathrm{MSO}_\Sigma[\tau]$ and, whenever $\mathrm{MSO}_\Sigma[\tau]$ is interpreted over trees or tree sequences, let $\mathrm{MPL}_\Sigma[\tau]$ be the restriction of $\mathrm{MSO}_\Sigma[\tau]$ obtained by constraining set variables to be interpreted over paths (formulas in $\mathrm{MPL}_\Sigma[\tau]$ are called *path formulas*).

Let $\varphi(x_1, \ldots, x_n, X_1, \ldots, X_m)$ be a formula with free individual variables $x_1, \ldots, x_n$ and free set variables $X_1, \ldots, X_m$. A sentence φ is a formula devoid of free variables. A *model* for a sentence φ is a structure in which φ is true. Let $\mathcal{S}$ be the set of structures over which $\mathrm{MFO}_\Sigma[\tau]$ (resp. $\mathrm{MPL}_\Sigma[\tau]$, $\mathrm{MSO}_\Sigma[\tau]$) is interpreted. We say that $T \subseteq \mathcal{S}$ is *definable* in $\mathrm{MFO}_\Sigma[\tau]$ (resp. $\mathrm{MPL}_\Sigma[\tau]$, $\mathrm{MSO}_\Sigma[\tau]$) if there exists a *sentence* φ in $\mathrm{MFO}_\Sigma[\tau]$ (resp. $\mathrm{MPL}_\Sigma[\tau]$, $\mathrm{MSO}_\Sigma[\tau]$) such that, for every $\mathcal{M} \in \mathcal{S}$, $\mathcal{M}$ is a model of φ if and only if $\mathcal{M} \in T$. Whenever no confusion can arise, we will omit the subscript $_\Sigma$. In the rest of the paper, we will focus our attention on the following languages and theories:

- $\mathrm{MFO}[<]$, interpreted over finite and infinite sequences;
- $\mathrm{MPL}[<, (\downarrow_i)_{i=0}^{k-1}]$, interpreted over infinite k-ary trees;
- $\mathrm{MFO/MPL/MSO}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$ and $\mathrm{MFO/MPL/MSO}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$, interpreted over infinite sequences of infinite k-ary trees.

It is possible to show that $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$ (resp. $\mathrm{MSO}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$) is as expressive as $\mathrm{MPL}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$ (resp. $\mathrm{MSO}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$).

PROPOSITION 3.2
$\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$, interpreted over infinite sequences of infinite k-ary trees, is as expressive as $\mathrm{MPL}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$.

PROOF. We first show that $(i,x) <_{tot} (j,y)$ if and only if either $i = j$ and $x <_{lex} y$ or $i < j$. Consider the direction from left to right. For the sake of conciseness, we use $(i,x)\overline{<}_{tot}(j,y)$ as a shorthand for either $i = j$ and $x <_{lex} y$ or $i < j$. In order to prove the implication, it suffices to show that axioms (1,2,3) and rules (4,5) of the definition of $<_{tot}$ (over TS_k) are *sound*, that is, they preserve the relation $\overline{<}_{tot}$.

1. $(i,\epsilon)\overline{<}_{tot}(j,\epsilon)$, with $0 \leq i < j$: it immediately follows from $i < j$.
2. $(i,x)\overline{<}_{tot} \downarrow_j (i,x)$, for $0 \leq j \leq k-1$: it follows from $\downarrow_j (i,x) = (i, \downarrow_j (x))$, $\downarrow_j (x) = xj$, and $x <_{lex} xj$.
3. $\downarrow_j (i,x)\overline{<}_{tot} \downarrow_{j+1} (i,x)$, for $0 \leq j \leq k-2$ and $i \geq 0$: it is analogous to the previous case.
4. if $(i,x)\overline{<}_{tot}(j,y)$ and either $i < j$ or it is not the case that $x < y$ (over T_k), then $\downarrow_{k-1} (i,x)\overline{<}_{tot}(j,y)$, for all $i,j \geq 0$. In the case in which $i < j$, then $\downarrow_{k-1} (i,x) = (i, \downarrow_{k-1} (x))\overline{<}_{tot}(j,y)$, and thus the thesis. Suppose that $i = j$ and $x <_{lex} y$. Since $x < y$ does not hold, by definition of $<_{lex}$, there exist a, b, z, u, and v, with $|a| = |b| = 1$ and $|z|, |u|, |v| \geq 0$, such that $x = zau$, $y = zbv$, and $0 \leq a < b$. Therefore $\downarrow_{k-1} (x) = x(k-1) = zau(k-1) = zau'$, with $u' = u(k-1)$. This allows us to conclude that $\downarrow_{k-1} (x) <_{lex} y$, and thus $\downarrow_{k-1} (i,x)\overline{<}_{tot}(j,y)$.
5. if $(i,x)\overline{<}_{tot}(n,z)$ and $(n,z)\overline{<}_{tot}(j,y)$, then $(i,x)\overline{<}_{tot}(j,y)$: a simple case analysis suffices, taking into account that both $<_{lex}$ and $<_1$ are transitive relations.

Let us prove now the opposite direction, that is, for any given i, j, x, and y, if either $i = j$ and $x <_{lex} y$ or $i < j$, then $(i,x) <_{tot} (j,y)$. Let $i < j$. For all $z \in T_k$, it holds that $(i,z) <_{tot} (j,\epsilon)$. The proof is by induction on the length of z. If $|z| = 0$, that is, $z = \epsilon$, the thesis follows from axiom (1). Let $|z| = n+1$. By definition, there exist w, with $|w| = n$ and m, with $0 \leq m \leq k-1$, such that $z = \downarrow_m (i,w)$. From axiom (3) and rule (5), it holds that $\downarrow_m (i,w) \leq_{tot} \downarrow_{k-1} (i,w)$. Furthermore, by the inductive hypothesis, it holds that $(i,w) <_{tot} (j,\epsilon)$, and thus, by using rule (4), we obtain that $\downarrow_{k-1} (i,w) <_{tot} (j,\epsilon)$. An application of rule (5) allows us to conclude that $(i,z) <_{tot} (j,\epsilon)$. If $y = \epsilon$ we have the thesis. Otherwise, we need to show that $(j,\epsilon) <_{tot} (j,y)$, for all y, with $|y| > 0$. This can be proved by a simple induction on the length of y. Given $(i,z) <_{tot} (j,\epsilon)$ and $(j,\epsilon) <_{tot} (j,y)$, rule (5) allows us to conclude that $(i,z) <_{tot} (j,y)$. The case in which $i = j$ and $x <_{lex} y$ can be dealt with in a similar way.

To complete the proof, it suffices to show that the relation $<_{tot}$ can be expressed in $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$ and that both $<_1$ and $<_2$ can be expressed in $\mathrm{MPL}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$. As for the first statement, we just showed that $(i,x) <_{tot} (j,y)$ if and only if either $i = j$ and $x <_{lex} y$ or $i < j$. On the ground of this equivalence, it is not difficult

to see that $x \leq_{tot} y$ if and only if

$$x \leq_2 y \vee \exists z(\bigvee_{0 \leq i < j \leq k} (\downarrow_i (z) \leq_2 x \wedge \downarrow_j (z) \leq_2 y)) \vee x \leq_1 y.$$

By exploiting the same correspondence, we can also prove the second statement. For all pairs x, y, $x \leq_1 y$ if and only if

$$x \leq_{tot} y \wedge \exists X \exists Y \exists r_1 \exists r_2 (r_1 \neq r_2 \wedge \mathtt{T}^0(r_1) \wedge \mathtt{T}^0(r_2) \wedge x \in X \wedge r_1 \in X \wedge y \in Y \wedge r_2 \in Y),$$

where $\mathtt{T}^0(x) =_{def} \neg \exists y \bigvee_{i=0}^{k-1} \downarrow_i (y) = x$.
Furthermore, for all pairs x, y, $x \leq_2 y$ if and only if

$$x \leq_{tot} y \wedge \exists X (x \in X \wedge y \in X)$$

■

The above proposition can be easily generalized to the full second-order case. One direction of the proof remains the same ($<_{tot}$ can be defined in $\mathrm{MSO}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$ in exactly the same way). To prove the opposite direction, it suffices to show that paths can be defined in $\mathrm{MSO}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$:

$$\begin{aligned} \mathtt{Path}(X) \quad =_{def} \quad & \exists x (T^0(x) \wedge x \in X \wedge \forall y ((T^0(y) \wedge x \neq y) \rightarrow y \notin X)) \wedge \\ & \forall y (y \in X \rightarrow \textstyle\bigvee_{j=0}^{k-1} (\downarrow_j (y) \in X \wedge \bigwedge_{i \neq j} \downarrow_i (y) \notin X)) \wedge \\ & \forall y (y \notin X \rightarrow \textstyle\bigwedge_{i=0}^{k-1} \downarrow_i (y) \notin X). \end{aligned}$$

3.3 Contextualization, displacement, and projection operators

In [23], Montanari et al. propose $\mathrm{MSO}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$ as the language for time granularity and define its interpretations over both upward and downward unbounded layered structures. In this section, we show that $\mathrm{MPL}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$, interpreted over downward unbounded layered structures, is expressive enough to capture the basic temporal operators of contextualization, local displacement, and projection (an informal account of these operators can be found in Section 1).

The case of projection is trivial, since the projection symbols $\downarrow_i$, for $0 \leq i \leq k-1$, belong to the language. As an example, the condition "p holds at the i-th projection of x" can be written as $\exists y (\downarrow_i (x) = y \wedge p(y))$. We can also define the converse relation of abstraction as follows. We say that x can be abstracted in y, notationally $\uparrow (x) = y$, if and only if it holds that $\bigvee_{i=0}^{k-1} \downarrow_i (y) = x$. Accordingly, the condition "p holds at the abstraction of x" can be written as $\exists y (\uparrow (x) = y \wedge p(y))$.

As for contextualization, let $\mathtt{T}^{\mathtt{i}}$ be a monadic predicate that holds at all time points of the i-th layer, for every $i \geq 0$ (this form of contextualization has been called horizontal contextualization in [23]). $\mathtt{T}^{\mathtt{i}}$ can be inductively defined as follows:

$$\begin{aligned} \mathtt{T}^{\mathtt{0}}(x) \quad & =_{def} \quad \neg \exists y \textstyle\bigvee_{i=0}^{k-1} \downarrow_i (y) = x; \\ \mathtt{T}^{\mathtt{i+1}}(x) \quad & =_{def} \quad \exists y (\mathtt{T}^{\mathtt{i}}(y) \wedge \textstyle\bigvee_{j=0}^{k-1} \downarrow_j (y) = x). \end{aligned}$$

Finally, local displacement is captured as follows. For every $i \geq 0$, let $+_{\mathtt{i}}\mathtt{1}$ be a binary predicate such that, for all pairs $x, y \in T^i$, $+_{\mathtt{i}}\mathtt{1}(\mathtt{x},\mathtt{y})$ if and only if y is the within T^i-successor of x. It can be defined as follows:

$$+_{\mathtt{i}}\mathtt{1}(x,y) \quad =_{def} \quad \mathtt{T^i}(x) \wedge \mathtt{T^i}(y) \wedge x <_{tot} y \wedge \forall z(\mathtt{T^i}(z) \wedge x <_{tot} z \rightarrow y \leq_{tot} z).$$

In the following, we will use a functional notation for $+_{\mathtt{i}}\mathtt{1}$, that is, we will write $+_{\mathtt{i}}\mathtt{1}(x) = y$ for $+_{\mathtt{i}}\mathtt{1}(x,y)$. Moreover, we will adopt $+_{\mathtt{i}}\mathtt{j}(x)$ as an abbreviation for $(+_{\mathtt{i}}\mathtt{1})^j(x)$.

In [23], Montanari et al. also introduce an alternative form of contextualization, called vertical contextualization, which is defined as follows: for every $i \geq 0$, let $\mathtt{D^i}$ be a monadic predicate that holds at all time points at distance i from the origin of the layer they belong to. We show that the combined use of projection, local displacement and horizontal contextualization makes it possible to define vertical contextualization. Given $w \in \{0,\ldots,k-1\}^*$, we inductively define $\downarrow_w (x)$ as follows. Whenever $|w| = 0$, $\downarrow_w (x) = x$. Otherwise, let $w = av$, with $a \in \{0,\ldots,k-1\}$. We define $\downarrow_{av} (x) = \downarrow_a (\downarrow_v (x))$. $\mathtt{D^0}(x)$ can be defined as follows:

$$\mathtt{D^0}(x) \quad =_{def} \quad \exists X(x \in X \wedge 0_0 \in X \wedge \forall y(y \in X \rightarrow \downarrow_0 (y) \in X)),$$

where 0_0 is the first-order definable origin of coarsest layer (we have that $y = 0_0$ if and only if $\forall z(y \leq_{tot} z)$)[1].

For all $i > 0$, let $a_n k^n + \ldots a_0 k^0$ be the k-ary representation of i. $\mathtt{D^i}$ can be defined as follows:

$$\mathtt{D^i}(x) \quad =_{def} \quad \bigvee_{j=0}^{\lfloor \log_k(i) \rfloor} \exists z(\mathtt{D^0}(z) \wedge \mathtt{T^j}(z) \wedge +_{\mathtt{j}}\mathtt{i}(z) = x) \vee \exists y(\mathtt{D^0}(y) \wedge \downarrow_{a_0,\ldots,a_n} (y) = x).$$

Since second-order quantification (over paths) only occurs in the definition $\mathtt{D^0}$, it immediately follows that horizontal contextualization, local displacement, and projection are also definable in $\mathrm{MFO}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$.

REMARK 3.3
In $\mathrm{MSO}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$, there is no way to define a binary predicate $\mathtt{EqualT}$ (resp. $\mathtt{EqualD}$) such that, for all x, y, $\mathtt{EqualT}(x,y)$ (resp. $\mathtt{EqualD}(x,y)$) holds true if and only if there exists i such that both $T^i(x)$ (resp. $D^i(x)$) and $T^i(y)$ (resp. $D^i(y)$) hold. Indeed, the addition of $\mathtt{EqualT}$ (resp. $\mathtt{EqualD}$) makes $\mathrm{MSO}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$ undecidable, a result which is closely related to the undecidability of the extension of $S2S$ with a predicate $\mathtt{Elevel}$ such that $\mathtt{Elevel}(x,y)$ holds true if and only if $|x| = |y|$, with $x, y \in \{0,1\}^*$ [19]. Notice that, from the fact that $\mathtt{EqualT}$ and $\mathtt{EqualD}$ cannot be expressed in $\mathrm{MSO}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$, it obviously follows that they cannot be expressed in $\mathrm{MPL}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$ and $\mathrm{MFO}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$, but it does not follow that the addition of these predicates to such theories, or to weaker variants of them devoid of the

[1] In order to define $\mathtt{D^0}$ in $\mathrm{MSO}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$, it suffices to add the conjunct $\mathtt{Path}(X)$ to the above definition, that is:

$$\mathtt{D^0}(x) \quad =_{def} \quad \exists X(\mathtt{Path}(X) \wedge x \in X \wedge 0_0 \in X \wedge \forall y(y \in X \rightarrow \downarrow_0 (y) \in X)).$$

uninterpreted monadic predicates P_a, with $a \in \Sigma$, would make them undecidable. We are currently exploring these decidability problems in a systematic way.

In the following, we will refer to $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$, which has been proved to be as expressive as $\mathrm{MPL}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$ (cf. Proposition 3.2). The reason is that, according to the *combining logic perspective*, $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$ allows us to specify the behaviour of a (granular) reactive system as a suitable combination of temporal evolutions, modeled by $<_1$, and temporal refinements, modeled by $<_2$ (cf. Figure 3).

We conclude by pointing out that the expressive power of $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$ does not change if we further constrain set variables to only range over full paths (instead of paths): a full path is just a particular path starting from a point belonging to the first temporal domain; moreover, a path can always be embedded into a unique full path. For instance, the sentence "there is a path whose time points are labeled with symbol a", which is expressed in $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$ by the formula:

$$\exists X \forall x (x \in X \ \rightarrow \ P_a(x)),$$

can be captured using full paths by the formula:

$$\exists X \exists y (y \in X \wedge \forall x (x \in X \ \wedge \ y \leq_2 x \ \rightarrow \ P_a(x))).$$

3.4 The relationships with interval structures and theories

There exists a natural link between structures and theories of time granularity and those developed for representing and reasoning about time intervals. Differently-grained temporal domains can indeed be interpreted as different ways of partitioning a given discrete/dense time axis into consecutive disjoint intervals. According to this interpretation, every time point can be viewed as a suitable interval over the time axis and projection implements an intervals/subintervals mapping. More precisely, let us define *direct constituents* of a time point x, belonging to a given domain, the time points of the immediately finer domain into which x can be refined (if any) and *indirect constituents* the time points into which the direct constituents of x can be directly or indirectly refined (if any). The mapping of a given time point into its direct or indirect constituents can be viewed as a mapping of a given time interval into (a subset of) its subintervals. It is worth investigating the possibility of lifting this mapping between interval and granularity structures to the corresponding theories. Most interval temporal logics, such as, for instance, Moszkowski's Interval Temporal Logic (ITL), Halpern and Shoham's xxx (HS), and xxx Neighboorhoud Logic (NL), have indeed been shown to be undecidable, unless we restrict ourselves to very weak fragments of them, e.g.

goal: transfer of decidability results from the granularity setting to the interval one; (one of the) key points: the restriction on the set of projection

4 Temporal logics for time granularity

In this section, we first give the definition of basic linear and branching time logics as well as that of their past and/or directed variants; then, we introduce temporal logics for time granularity.

Let $\mathcal{P}_\Sigma$ ($\mathcal{P}$ when no confusion can arise) be the finite set of atomic propositional letters $\{P_a \mid a \in \Sigma\}$.

DEFINITION 4.1
((Past) Directed CTL^* and (Past) Directed PTL)

We inductively define a set of *state* formulas and a set of *path* formulas:

- state formulas
  (S1) any atomic proposition $P_a \in \mathcal{P}_\Sigma$ is a state formula;
  (S2) if p, q are state formulas, then $p \wedge q$ and $\neg p$ are state formulas;
  (S3) if p is a path formula, then $\mathbf{A}p$ and $\mathbf{E}p$ are state formulas;
- path formulas
  (P0) any atomic proposition $P_a \in \mathcal{P}_\Sigma$ is a path formula;
  (P1) any state formula is a path formula;
  (P2) if p, q are path formulas, then $p \wedge q$ and $\neg p$ are path formulas;
  (P3) if p, q are path formulas, then $\mathbf{X}p$, and $p\mathbf{U}q$ are path formulas;
  (P4) if p is a path formula, then $\mathbf{X_0}p, \ldots, \mathbf{X_{k-1}}p$ are path formulas;
  (P5) if p, q are path formulas, then $\mathbf{X^{-1}}p$ and $p\mathbf{S}q$ are path formulas.

As for branching time logics, the language of *Past (k-ary) Directed* CTL^* ($\mathrm{PCTL}^*_\mathrm{k}$ for short) is the smallest set of state formulas generated by the above rules. The language of *(k-ary) Directed* CTL^* ($\mathrm{CTL}^*_\mathrm{k}$) is obtained by eliminating rule (P5), that of *Past* CTL^* (PCTL^*) by removing rule (P4), and that of CTL^* by deleting both (P4) and (P5).
As for linear time logics, the language of *Past (k-ary) Directed* PTL ($\mathrm{PPTL_k}$) is the smallest set of path formulas generated by the rules (P0), (P2), (P3), (P4), and (P5). The language of *(k-ary) Directed* PTL ($\mathrm{PTL_k}$) is obtained by deleting rule (P5), that of *Past* PTL (PPTL) by eliminating rule (P4), and that of PTL by deleting both (P4) and (P5). Formulas $\mathbf{F}p$ and $\mathbf{G}p$ are respectively defined as $\mathtt{true}\mathbf{U}p$ and $\neg\mathbf{F}\neg p$ as usual, where $\mathtt{true} = P_a \vee \neg P_a$, for some $P_a \in \mathcal{P}_\Sigma$.

We interpret (P)PTL over sequences (or unary trees), (P)$\mathrm{PTL_k}$ over paths of k-ary trees (with $k \geq 2$), and (PD)$\mathrm{CTL}^*_\mathrm{k}$ over k-ary trees (with $k \geq 2$). The semantic interpretation for non-directed logics is given as usual [10]. The semantic interpretation for (P)$\mathrm{CTL}^*_\mathrm{k}$ coincides with that for (P)CTL^*, except for path formulas of the form $\mathbf{X_i}p$, whose interpretation is defined as follows. Given $t \in T_k(\Sigma)$, a path X in t, and a position j in X,

$$t, X, j \models \mathbf{X_i}p \text{ if and only if } X(j+1) = \downarrow_i (X(j)) \text{ and } t, X, j+1 \models p.$$

In the following, we will make use of operators $\mathbf{N_i}p$ defined as $\mathbf{EX_i}p$, for all $i = 0, \ldots, k-1$.

In a similar way, the semantic interpretation for (P)$\mathrm{PTL_k}$ coincides with that for (P)PTL, except for formulas of the form $\mathbf{X_i}p$, whose interpretation is defined as follows. Given a (full) path X in a k-ary tree and a position j in X,

$$X, j \models \mathbf{X_i}p \text{ if and only if } X(j+1) =\downarrow_i (X(j)) \text{ and } X, j+1 \models p.$$

Let us now define the Computational Tree Sequence Logic with k successors ($\mathrm{CTSL_k^*}$), together with its past variant $\mathrm{PCTSL_k^*}$. Let p be a (P)$\mathrm{CTL_k^*}$-formula. We call p *monolithic* if its outermost operator is not a Boolean connective. For example, $\mathbf{AG}(P \vee Q)$ is monolithic, while $\mathbf{AG}P \vee Q$ is not. Formulas of (P)$\mathrm{CTSL_k^*}$ are obtained by embedding monolithic formulas of (P)$\mathrm{CTL_k^*}$ into (P)PTL. To avoid confusion, we rename the linear temporal operators $\mathbf{X}$, $\mathbf{U}$, $\mathbf{X}^{-1}$, and $\mathbf{S}$ of (P)PTL by $\bigcirc$, $\triangle$, $\bigcirc^{-1}$, and $\triangle^{-1}$, respectively.

DEFINITION 4.2
($\mathrm{CTSL_k^*}$ and $\mathrm{PCTSL_k^*}$)

We inductively define a set of *layered* formulas:

(L0) any monolithic formula in $\mathrm{CTL_k^*}$ is a layered formula;

(L1) any monolithic formula in $\mathrm{PCTL_k^*}$ is a layered formula;

(L2) if p, q are layered formulas, then $p \wedge q$ and $\neg p$ are layered formulas;

(L3) if p, q are layered formulas, then $\bigcirc p$ and $p \triangle q$ are layered formulas;

(L4) if p, q are layered formulas, then $\bigcirc^{-1} p$ and $p \triangle^{-1} q$ are layered formulas.

The language of $\mathrm{PCTSL_k^*}$ is the smallest set of formulas generated by rules (L1), (L2), (L3), and (L4), while that of $\mathrm{CTSL_k^*}$ is obtained by applying rules (L0), (L2), and (L3). Formulas $\Diamond p$ and $\Box p$ are respectively defined as $\mathtt{true} \triangle p$ and $\neg\Diamond\neg p$ as usual.

For every $k \geq 2$, we interpret (P)$\mathrm{CTSL_k^*}$ formulas over k-ary tree sequences as follows. Given a tree sequence $ts \in TS_k(\Sigma)$, linear temporal operators $\bigcirc$, $\triangle$, $\bigcirc^{-1}$, and $\triangle^{-1}$ are interpreted over the first layer L_0 of ts, while branching temporal operators $\mathbf{E}$ and $\mathbf{A}$ are interpreted over trees rooted at time points in L_0. Given a tree sequence $ts \in TS_k(\Sigma)$, a position $j \geq 0$ in L_0, and a (P)$\mathrm{CTSL_k^*}$-formula p, we define the satisfiability relation $ts, L_0, j \models p$ in terms of the satisfiability relation of (P)$\mathrm{CTL_k^*}$ (here denoted, to avoid confusion, by $\models_{\mathrm{CTL}}$) as follows:

$$
\begin{array}{lll}
ts, L_0, j \models p & \text{iff} & ts_{L_0(j)}, L_0(j) \models_{\mathrm{CTL}} p,\ p \text{ monolithic in } (P)\mathrm{CTL}^*_{\mathrm{k}} \\
ts, L_0, j \models p \wedge q & \text{iff} & ts, L_0, j \models p \text{ and } ts, L_0, j \models q; \\
ts, L_0, j \models \neg p & \text{iff} & \text{it is not the case that } ts, L_0, j \models p; \\
ts, L_0, j \models \bigcirc p & \text{iff} & ts, L_0, j+1 \models p; \\
ts, L_0, j \models p \triangle q & \text{iff} & ts, L_0, r \models q \text{ for some } r \geq j, \text{ and} \\
 & & ts, L_0, s \models p \text{ for every } j \leq s < r; \\
ts, L_0, j \models \bigcirc^{-1} p & \text{iff} & j > 0 \text{ and } ts, L_0, j-1 \models p; \\
ts, L_0, j \models p \triangle^{-1} q & \text{iff} & ts, L_0, r \models q \text{ for some } r \leq j, \text{ and} \\
 & & ts, L_0, s \models p \text{ for every } r < s \leq j.
\end{array}
$$

Given a tree sequence $ts \in TS_k(\Sigma)$ and a (P)$\mathrm{CTSL}^*_{\mathrm{k}}$-formula p, we say that ts is a *model* of p if and only if $ts, L_0, 0 \models p$. A set $T \subseteq TS_k(\Sigma)$ is *definable* in (P)$\mathrm{CTSL}^*_{\mathrm{k}}$ if there exists a formula p in (P)$\mathrm{CTSL}^*_{\mathrm{k}}$ such that, for every $ts \in TS_k(\Sigma)$, ts is a model of p if and only if $ts \in T$.

5 Expressiveness and Complexity of $\mathrm{CTSL}^*_{\mathrm{k}}$

In this section, we show that $\mathrm{CTSL}^*_{\mathrm{k}}$ is as expressive as $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$, when interpreted over k-ary downward unbounded layered structures or, equivalently, over infinite sequences of infinite k-ary trees (Theorems 5.7 and 5.15). Furthermore, we prove that its satisfiability problem is 2EXPTIME-complete (Theorem 5.16).

5.1 Preliminaries

We start by recalling some basic definitions and properties of linear and branching time logics, and by stating some auxiliary results about branching time logics with explicit successors. Given two languages $\mathcal{L}_1$ and $\mathcal{L}_2$, interpreted over the same class $\mathcal{S}$ of structures, we say that $\mathcal{L}_1$ is *as expressive as* $\mathcal{L}_2$ if, for every set $T \subseteq \mathcal{S}$, T is $\mathcal{L}_1$-definable if and only if T is $\mathcal{L}_2$-definable.

As for linear time logic, it is well-known that, when interpreted over the class of finite sequences as well as over the class of infinite ones, PTL and PPTL are as expressive as $\mathrm{MFO}[<]$ [16, 18].

THEOREM 5.1
(Expressive completeness of PTL and PPTL)

PTL and PPTL are as expressive as $\mathrm{MFO}[<]$, when interpreted over infinite (resp. finite) sequences.

In [29], Sistla and Clarke show that the satisfiability problem for PTL is PSPACE-complete.

As for branching time logic, the expressive power of CTL^* and PCTL^* is equivalent to the one of monadic second-order logic on infinite binary trees with second-order quantifiers over infinite paths [17].

THEOREM 5.2
(Expressive completeness of CTL^* and PCTL^*)

CTL^* and PCTL^* are as expressive as $\mathrm{MPL}[<]$, when interpreted over infinite binary trees.

In [7], Emerson and Jutla prove that the problem of testing satisfiability for CTL^* is 2EXPTIME-complete.

As pointed out by Hafer and Thomas [17], Theorem 5.2 can be generalized to $\mathrm{CTL}^*_{\mathrm{k}}$ and $\mathrm{PCTL}^*_{\mathrm{k}}$ with respect to $\mathrm{MPL}[<, (\downarrow_i)_{i=0}^{k-1}]$ by incorporating successors into both temporal and monadic path logics [17].

THEOREM 5.3
(Expressive completeness of $\mathrm{CTL}^*_{\mathrm{k}}$ and $\mathrm{PCTL}^*_{\mathrm{k}}$)

$\mathrm{CTL}^*_{\mathrm{k}}$ and $\mathrm{PCTL}^*_{\mathrm{k}}$ are as expressive as $\mathrm{MPL}[<, (\downarrow_i)_{i=0}^{k-1}]$, when interpreted over infinite k-ary trees.

Furthermore, a decision procedure for $\mathrm{CTL}^*_{\mathrm{k}}$ can be obtained by means of the following non trivial adaptation of the decision procedure for CTL^* originally developed by Emerson and Sistla [9] and later refined by Emerson and Jutla [7].

Let us assume $k = 2$ (the generalization to an arbitrary k is straightforward). As a preliminary step, we provide an embedding of PTL_2 into PTL. To this end, we define a translation τ of formulas and models of PTL_2, over an alphabet Σ, to formulas and models of PTL, over an extended alphabet $\Sigma \times \{0, 1\}$. The mapping of PTL_2-formulas into PTL-formulas is defined as follows:

$$\begin{array}{lcl}
\tau(P_a) & = & \bigvee_{i\in\{0,1\}} P_{(a,i)} \text{ for } a \in \Sigma \\
\tau(\alpha \wedge \beta) & = & \tau(\alpha) \wedge \tau(\beta) \\
\tau(\neg\alpha) & = & \neg\tau(\alpha) \\
\tau(\mathbf{X_i}\alpha) & = & \bigvee_{x\in\Sigma} P_{(x,i)} \wedge \mathbf{X}\tau(\alpha) \text{ for } i \in \{0,1\} \\
\tau(\alpha\mathbf{U}\beta) & = & \tau(\alpha)\mathbf{U}\tau(\beta)
\end{array}$$

Temporal structures for PTL_2 over Σ can be viewed as infinite labeled (full) paths, whose nodes are labeled with symbols in Σ and whose transitions are labeled with symbols in $\{0, 1\}$, while temporal structures for PTL over $\Sigma \times \{0, 1\}$ can be viewed as infinite labeled sequences, whose nodes are labeled with symbols in $\Sigma \times \{0, 1\}$. We define a *bijection* τ from temporal structures $\mathcal{M}$ for PTL_2 over Σ to temporal structures $\mathcal{M}'$ for PTL over $\Sigma \times \{0, 1\}$ that maps each a-labeled node of $\mathcal{M}$, with an outgoing transition labeled with i, to an (a, i)-labeled node of $\mathcal{M}'$. As an example, the (full) path $\ldots a \rightarrow_0 b \rightarrow_1 a \rightarrow_1 \ldots$ is mapped into the sequence $\ldots (a, 0)(b, 1)(a, 1) \ldots$. The following lemma can be easily proved.

LEMMA 5.4
For every formula φ of PTL_2 and temporal structure $\mathcal{M}$, $\mathcal{M}$ is a model for φ if and only if $\tau(\mathcal{M})$ is a model for $\tau(\varphi)$.

As a second preliminary step, we transform $\mathrm{CTL}^*_{\mathrm{k}}$-formulas in a normal form suitable for subsequent manipulation. Such a normal form is a straightforward generalization of the normal form for CTL^*-formulas proposed by Emerson and Sistla [9]. This result is formally stated by the following lemma, whose proof is similar to the one for CTL^* and thus omitted.

LEMMA 5.5
For any given $\mathrm{CTL}^*_{\mathrm{k}}$-formula φ_0, there exists a corresponding formula φ_1 in a normal form composed of conjunctions and disjunctions of subformulas of the form $\mathbf{A}p$, $\mathbf{E}p$, and $\mathbf{AGE}p$, where p is a pure linear time formula of PTL_2, such that (i) φ_1 is satisfiable if and only if φ_0 is satisfiable, and (ii) $|\varphi_1| = \mathcal{O}(|\varphi_0|)$. Moreover, any model of φ_1 can be used to define a model of φ_0 and vice versa.

THEOREM 5.6
($\mathrm{CTL}^*_{\mathrm{k}}$ is elementarily decidable)

The satisfiability problem for $\mathrm{CTL}^*_{\mathrm{k}}$ is 2EXPTIME-complete.

PROOF. Hardness follows from 2EXPTIME-hardness of CTL^* [32]. To show that it belongs to 2EXPTIME, we outline an algorithm for checking satisfiability for $\mathrm{CTL}^*_{\mathrm{k}}$ of deterministic doubly exponential time complexity. Given a $\mathrm{CTL}^*_{\mathrm{k}}$-formula φ_0, such an algorithm can be obtained as follows:

1. by exploiting Lemma 5.5, construct an equivalent formula φ_1 composed of conjunctions and disjunctions of subformulas of the form $\mathbf{A}p$, $\mathbf{E}p$, and $\mathbf{AGE}p$, where p is a PTL_2-formula;
2. by exploiting Lemma 5.4, replace every maximal PTL_2-formula p (over Σ) in φ_1 by the PTL-formula $\tau(p)$ (over $\Sigma \times \{0,1\}$); then, construct a Büchi string automaton $\mathcal{A}_{\tau(p)}$ (over $\Sigma \times \{0,1\}$) recognizing the models of $\tau(p)$, by using the technique described in [9];
3. for every subformula of the form $\mathbf{A}p$ of φ_1, determinize the Büchi string automaton $\mathcal{A}_{\tau(p)}$ for $\tau(p)$, using Safra's construction [28], to obtain an equivalent deterministic Rabin string automaton $\mathcal{A}'_{\tau(p)}$ (over $\Sigma \times \{0,1\}$) for $\tau(p)$;
4. program a Rabin tree automaton $\mathcal{A}_{\varphi_1}$ (over Σ), accepting the models of φ_1, which incorporates the string automata built in steps 2 and 3 in a suitable way (see below);
5. test whether $\mathcal{A}_{\varphi_1}$ recognizes the empty language using the algorithm proposed by Emerson and Jutla [7].

Step 4 is as follows. For every subformula $\mathbf{E}p$ of φ_1, let $\mathcal{A}_p = (Q, q_0, \Delta, F)$ be the Büchi string automaton for p. We construct the Büchi tree automaton $\mathcal{A}_{\mathbf{E}p} = (Q \cup \{q_*\}, q_0, \Delta', F \cup \{q_*\})$ for $\mathbf{E}p$, where Δ' is defined as follows:

$$\begin{array}{lll} \Delta'(q,a,q',q_*) & \text{if and only if} & \Delta(q,(a,0),q'); \\ \Delta'(q,a,q_*,q') & \text{if and only if} & \Delta(q,(a,1),q'); \\ \Delta'(q_*,a,q_*,q_*) & \text{if and only if} & a \in \Sigma. \end{array}$$

For every subformula $\mathbf{A}p$ of φ_1, let $\mathcal{A}_p = (Q, q_0, \Delta, \Gamma)$ be the deterministic Rabin string automaton for p. We construct the deterministic Rabin tree automaton $\mathcal{A}_{\mathbf{A}p} = (Q, q_0, \Delta', \Gamma)$ for $\mathbf{A}p$, where Δ' is defined as follows:

$$\Delta'(q, a, q', q'') \quad \text{if and only if} \quad \Delta(q, (a, 0), q') \text{ and } \Delta(q, (a, 1), q'').$$

For every subformula $\mathbf{AGE}p$ of φ_1, let $\mathcal{A}_p = (Q, q_0, \Delta, F)$ be the Büchi string automaton for p. We construct the Büchi tree automaton $\mathcal{A}_{\mathbf{AGE}p} = (Q \cup \{q^* \mid q \in Q\}, q_0, \Delta', F \cup \{q^* \mid q \in Q\})$ for $\mathbf{AGE}p$, where Δ' is defined as follows:

$$\begin{array}{lll}
\Delta'(q, a, q', q_1^*) & \text{if and only if} & \Delta(q, (a, 0), q') \text{ and } \Delta(q_0, (a, 1), q_1); \\
\Delta'(q, a, q_1^*, q') & \text{if and only if} & \Delta(q, (a, 1), q') \text{ and } \Delta(q_0, (a, 0), q_1); \\
\Delta'(q, a, q', q_0^*) & \text{if and only if} & \Delta(q, (a, 0), q') \text{ and } \Delta(q_0, (a, 0), q'); \\
\Delta'(q, a, q_0^*, q') & \text{if and only if} & \Delta(q, (a, 1), q') \text{ and } \Delta(q_0, (a, 1), q'); \\
\Delta'(q^*, a, q', q'') & \text{if and only if} & \Delta'(q, a, q', q'').
\end{array}$$

A tree automaton $\mathcal{A}_{\varphi_1}$ for φ_1 is obtained by taking the intersection and/or the union of the tree automata constructed for the subformulas of φ_1. Notice that if φ_1 does not contain subformulas of the form $\mathbf{X_i}p$, then $\mathcal{A}_{\varphi_1}$ is symmetric.

By exploiting classical complexity results on string and tree automata [10, 30, 31], an easy complexity analysis of the proposed algorithm allows us to conclude that its complexity is (deterministic) 2EXPTIME. ∎

5.2 *Expressive completeness of* $\mathrm{CTSL}_{\mathrm{k}}^*$

We prove that $\mathrm{CTSL}_{\mathrm{k}}^*$ is as expressive as $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$, when interpreted over infinite sequences of infinite k-ary trees.

For technical reasons, we constrain path variables X to be interpreted over full paths. As shown in Section 3, such a restriction leaves the expressive power of monadic path formulas unchanged. For the same reason, we reformulate the considered monadic languages as follows (without altering their expressive power). Given the vocabularies $\tau_1 = \{\epsilon, <_1, <_2, (\downarrow_i)_{i=0}^{k-1}\}$, $\tau_2 = \{\epsilon, <, (\downarrow_i)_{i=0}^{k-1}\}$, $\tau_3 = \{\epsilon, <, \downarrow\}$ and $\tau_4 = \{\epsilon, \delta, <, \downarrow\}$, we will use:

- $\mathrm{MPL}[\tau_1]$, interpreted over infinite sequences of infinite k-ary trees, where ϵ is a (first-order definable) constant denoting the root of the first tree of the sequence;
- $\mathrm{MPL}[\tau_2]$, interpreted over infinite k-ary trees, where ϵ is a (first-order definable) constant denoting the root of the tree;
- $\mathrm{MFO}[\tau_4]$ (resp. $\mathrm{MFO}[\tau_3]$), interpreted over finite (resp. infinite) sequences, where ϵ and δ are constants that respectively denote the first and the last element of the sequence, and $\downarrow$ is the successor function. It is easy to show that ϵ and δ, as well as $\downarrow$, are (first-order) definable in terms of $<$.

One direction of the proof (as usual) is easy: there exists a straightforward embedding of $\mathrm{CTSL}_{\mathrm{k}}^*$ into $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$.

THEOREM 5.7
($\mathrm{CTSL}^*_{\mathrm{k}}$ is a fragment of $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$)

For every set $T \subseteq TS_k(\Sigma)$ of infinite sequences of Σ-labeled infinite trees, if T is definable in $\mathrm{CTSL}^*_{\mathrm{k}}$, then T is definable in $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$.

PROOF. We prove that every $\mathrm{CTSL}^*_{\mathrm{k}}$-formula can be mapped into an equivalent $\mathrm{MPL}[\tau_1]$-sentence, and, thus, that it is equivalent to a $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$-sentence. To this end, for any $ts \in TS_k(\Sigma)$, $w \in ts$, and $p \in \mathrm{CTSL}^*_{\mathrm{k}}$, we define a translation τ_w of p into an $\mathrm{MPL}[\tau_1]$-formula, with free variable w. For any $\mathrm{CTL}^*_{\mathrm{k}}$-formula p, let $\hat{\tau}_w(p)$ be the $\mathrm{MPL}[\tau_2]$-formula obtained from p by exploiting Theorem 5.3, with the symbol $<$ replaced by $<_2$. The translation τ_w can be defined as follows:

$$
\begin{array}{lcl}
\tau_w(p) & = & \hat{\tau}_w(p), \text{ whenever } p \text{ is monolithic in } \mathrm{CTL}^*_{\mathrm{k}}; \\
\tau_w(p \wedge q) & = & \tau_w(p) \wedge \tau_w(q); \\
\tau_w(\neg p) & = & \neg \tau_w(p); \\
\tau_w(\bigcirc p) & = & \exists w'(w' = +_1^0(w) \wedge \tau_{w'}(p)); \\
\tau_w(p \triangle q) & = & \exists w'(T^0(w') \wedge w \leq_1 w' \wedge \tau_{w'}(q) \wedge \\
 & & \forall w''((T^0(w'') \wedge w \leq_1 w'' \wedge w'' <_1 w') \rightarrow \tau_{w''}(p))).
\end{array}
$$

For any $\mathrm{CTSL}^*_{\mathrm{k}}$-formula p, we have that p is equivalent to the $\mathrm{MPL}[\tau_1]$-sentence $\varphi_p = \tau_\epsilon(p)$. ∎

In order to prove the opposite direction, that is, to show that $\mathrm{CTSL}^*_{\mathrm{k}}$ is expressively complete with respect to $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$, we follow a 'decomposition method' similar to that exploited by Hafer and Thomas in [17] to prove the expressive completeness of CTL^* with respect to $\mathrm{MPL}[<]$. We first decompose the model checking problem for an $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$-formula and a tree sequence in $TS_k(\Sigma)$ into a *finite* number of model checking subproblems for formulas and structures that do not refer to the whole tree sequence anymore, but only to certain disjoint components of it. Then, taking advantage of such a decomposition step, we map every $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$-formula into an equivalent (but sometimes much longer) $\mathrm{CTSL}^*_{\mathrm{k}}$-formula.

As a preliminary step, we show that the addition of past operators to $\mathrm{CTSL}^*_{\mathrm{k}}$ does not increase its expressive power.

LEMMA 5.8
$\mathrm{PCTSL}^*_{\mathrm{k}}$ is as expressive as $\mathrm{CTSL}^*_{\mathrm{k}}$.

PROOF. By exploiting Theorem 5.3, any $\mathrm{PCTSL}^*_{\mathrm{k}}$-formula p can be replaced by an equivalent $\mathrm{PCTSL}^*_{\mathrm{k}}$-formula q, whose past operators (if any) are of the forms $\bigcirc^{-1}$ and $\triangle^{-1}$ only. Let $q_1, \ldots q_n$ be the monolithic $\mathrm{CTL}^*_{\mathrm{k}}$-subformulas of q. Regarding $q_1, \ldots q_n$ as additional atomic propositions within q, we may consider q as a PPTL-formula. By exploiting Kamp's Theorem, q can be replaced by an equivalent PTL-formula r that contains $q_1, \ldots q_n$ as subformulas. ∎

Let $\equiv_n$ be a relation on $TS_k(\Sigma)$ such that $ts \equiv_n ts'$ if and only if ts and ts' satisfy the same sentences of $\mathrm{MPL}[\tau_1]$ of quantifier depth n. It is possible to show that $\equiv_n$ is

an equivalence relation of *finite* index. Its equivalence classes are called n*-types* and are described by path formulas called n-types descriptors.

DEFINITION 5.9
(n-types descriptors)

Given $ts \in TS_k(\Sigma)$, a sequence of r elements $\overline{a}$ in ts, a sequence of s full paths $\overline{P}$ in ts, and $n \geq 0$, an n-type descriptor $\psi^n_{ts,\overline{a},\overline{P}}$ is a path formula defined as follows:

$$\begin{array}{lcl} \psi^0_{ts,\overline{a},\overline{P}} & = & \bigwedge\{\varphi(x_1 \dots x_r, X_1 \dots X_s) \mid \varphi \text{ atomic or negated atomic, } ts \models \varphi[\overline{a},\overline{P}]\} \\ \psi^{n+1}_{ts,\overline{a},\overline{P}} & = & \bigwedge_{a \in ts} \exists x_{r+1} \psi^n_{ts,\overline{a}a,\overline{P}} \wedge \bigvee_{a \in ts} \forall x_{r+1} \psi^n_{ts,\overline{a}a,\overline{P}} \wedge \\ & & \bigwedge_{P \subseteq \Pi(ts)} \exists X_{r+1} \psi^n_{ts,\overline{a},\overline{P}P} \wedge \bigvee_{P \subseteq \Pi(ts)} \forall X_{r+1} \psi^n_{ts,\overline{a},\overline{P}P} \end{array}$$

The relation $\equiv_n$ can be characterized by an Ehrenfeucht game $G_n(ts, ts')$ as follows (basics on Ehrenfeucht games can be found, for instance, in [8]). A play of this game is played by two players Spoiler and Duplicator on structures $ts, ts' \in TS_k(\Sigma)$ and consists of n rounds. At each round, Spoiler chooses an element or a full path either from ts or from ts'; Duplicator reacts by choosing an object of the same kind in the other structure. After n rounds, elements $a_1, \dots a_r$ ($\overline{a}$ for short) and full paths $P_1, \dots P_s$ ($\overline{P}$) in ts (with $n = r+s$), and the corresponding elements $b_1, \dots b_r$ ($\overline{b}$) and full paths $Q_1, \dots Q_s$ ($\overline{Q}$) in ts', have been chosen. Duplicator wins if the map $\overline{a} \to \overline{b}$ is a *partial isomorphism* from $(ts, \overline{P})$ to $(ts', \overline{Q})$, i.e., it is injective and respects ϵ, $<_2$, $<_1$, and $\downarrow_i$, with $i = 0, \dots k-1$, as well as membership in P_a, for every $a \in \Sigma$. The game can be naturally extended to $G_n((ts, \overline{P}), \overline{a}, (ts', \overline{Q}), \overline{b})$, where $\overline{a}$ and $\overline{P}$ (resp. $\overline{b}$ and $\overline{Q}$) are a finite sequence of elements and a finite sequence of full paths in ts (resp. in ts'), respectively.

Let $\sim_n$ be a relation such that, for any pair of structures $(ts, \overline{a}, \overline{P})$ and $(ts', \overline{b}, \overline{Q})$, $(ts, \overline{a}, \overline{P}) \sim_n (ts', \overline{b}, \overline{Q})$ if and only if Duplicator wins the game $G_n((ts, \overline{P}), \overline{a}, (ts', \overline{Q}), \overline{b})$. The following result easily follows from the well-known Ehrenfeucht-Fraïssé Theorem.

THEOREM 5.10
Given structures ts and ts', element sequences $\overline{a}$ in ts and $\overline{b}$ in ts', full path sequences $\overline{P}$ in ts and $\overline{Q}$ in ts', the following are equivalent conditions:

1. $(ts, \overline{a}, \overline{P}) \sim_n (ts', \overline{b}, \overline{Q})$;
2. $ts' \models \psi^n_{ts,\overline{a},\overline{P}}[\overline{b}, \overline{Q}]$;
3. $\overline{a}, \overline{P}$ satisfy in ts the same formulas of $\mathrm{MPL}[\tau_1]$ of quantifier depth less than or equal to n as $\overline{b}, \overline{Q}$ in ts'.

COROLLARY 5.11
Given $n \geq 0$ and an $\mathrm{MPL}[\tau_1]$-formula $\varphi(\overline{x}, \overline{X})$ of quantifier depth less than or equal to n, φ is equivalent to a *finite* disjunction of formulas $\psi^n_{ts,\overline{a},\overline{P}}$ such that $ts \models \varphi[\overline{a}, \overline{P}]$.

Similar definitions and results hold for k-ary tree structures and infinite (as well as finite) word structures. In the former case, n-type descriptors are path formulas in

the language of k-ary trees $\mathrm{MPL}[\tau_2]$. In the latter case, the rules of the game are that Spoiler and Duplicator can only pick *elements* from the given pair of words; hence, n-type descriptors are formulas in the *first-order* language of linear orders $\mathrm{MFO}[\tau_3]$ (resp. $\mathrm{MFO}[\tau_4]$).

Let $ts \in TS_k(\Sigma)$, k_0 be an element belonging to the i-th tree of ts, n be an integer greater than or equal to 0, and m be the index of $\equiv_n$. We enlarge the alphabet Σ to $\Sigma_1 = \Sigma \times \{1, \ldots, m\}$, where all $j \in \{1 \ldots m\}$ serve as indices for all possible n-types. Let π_1 (resp. π_3) be the finite (resp. infinite) sequence of roots $(\epsilon)_0, \ldots (\epsilon)_{i-1}$ (resp. $(\epsilon)_{i+1}, \ldots$). We denote by $v_1(ts, k_0)$ (resp. $v_3(ts, k_0)$) the finite (resp. infinite) word over Σ_1 whose l-th letter is (a, j) if $\pi_1(l)$ (resp. $\pi_3(l)$) carries letter $a \in \Sigma$ and the tree rooted in it has n-type j. Similarly, we enlarge the alphabet Σ to $\Sigma_2 = \Sigma \times (\{0, \ldots, k-1\} \times \{1, \ldots, m\})^{k-1}$, and we denote by π_2 the finite path from $(\epsilon)_i$ up to (and excluding) k_0. We denote by $v_2(ts, k_0)$ the finite word over Σ_2 whose l-th letter is $(a, (l_1, j_1), \ldots, (l_{k-1}, j_{k-1}))$ if $\pi_2(l)$ carries letter $a \in \Sigma$ and, for $r = 1, \ldots, k-1$, the tree rooted in the l_r-th son of it has n-type j_r.

We need to prove the following auxiliary lemma, which states that combining *local* winning strategies on disjoint parts of two tree sequences it is possible to obtain a *global* winning strategy on the two tree sequences.

LEMMA 5.12
For arbitrary (ts, k_0) and (ts', k_0'), if $v_i(ts, k_0) \sim_n v_i(ts', k_0')$, with $i = 1, 2, 3$, $ts_{k_0,i} \sim_n ts'_{k_0,i}$, with $i = 0, \ldots, k-1$, and $ts(k_0) = ts'(k_0')$, then $(ts, k_0) \sim_n (ts', k_0')$.

PROOF. Suppose that Spoiler picks an element k (different from k_0) in ts. If k belongs to some path $v_i(ts, k_0)$ or to some $ts_{k_0,i}$, then Duplicator chooses k' in ts' according to the corresponding local winning strategies. If k belongs to the tree k_0 belongs to, and neither $k <_2 k_0$ nor $k_0 <_2 k$, then Duplicator looks for the last node k_1 belonging to the path $v_2(ts, k_0)$ such that $k_1 <_2 k$. Let k_2 be the son of k_1 such that there is a path from k_2 to k. Suppose that k_2 is the i-th son of k_1. Duplicator chooses k_1' in the path $v_2(ts', k_0')$ according to his winning strategy on $v_2(ts, k_0), v_2(ts', k_0')$. Let k_2' be the i-th son of k_1'. Hence, k_1 and k_1' have the same label from Σ_2 and thus the subtrees rooted in their successors k_2 and k_2' have the same n-type. Hence, by Theorem 5.10 (its variant for k-ary trees), Duplicator has a winning strategy on these subtrees and can use this strategy to choose an element k' corresponding to k in the subtree rooted in k_2'. Finally, if $k <_1 k_0$ (resp. $k_0 <_1 k$) and k is not a root, then Duplicator chooses k' in ts' exploiting his winning strategy on $v_1(ts, k_0), v_1(ts', k_0')$ (resp. $v_3(ts, k_0), v_3(ts', k_0')$).

Suppose now that Spoiler picks a full path P_0 in ts. Let A (resp. A') be the finite path from the root of the tree k_0 (resp. k_0') belongs to up to (and excluding) k_0 (resp. k_0'). If $k_0 \in P_0$, then P_0 has the form A, k_0, B, where B is a path on $ts_{k_0,i}$, for some i. Hence, Duplicator chooses a path B' in $ts'_{k_0',i}$ according to his local strategy on $ts_{k_0,i}$, $ts'_{k_0',i}$, and he responds to Spoiler with the full path A', k_0', B'. If $k_0 \notin P_0$ and P_0 belongs to the tree k_0 belongs to, then $P_0 = A_1 k_1 C$, where k_1 is the last node of

P_0 which belongs to A and C is a path on $ts_{k_1,i}$, for some i. Duplicator first chooses k_1' according to his local strategy on $v_2(ts, k_0), v_2(ts', k_0')$. Let A_1' be the finite path from the root of the tree k_0' belongs to up to (and excluding) k_1'. Then, Duplicator picks a path C' on $ts'_{k_1',i}$ exploiting his winning strategy on $ts_{k_1,i}$, $ts'_{k_1',i}$, and, finally, he responds to Spoiler with the full path A_1', k_1', C' . The case in which P_0 does not belong to the tree k_0 belongs to is similar to the previous one, and thus its analysis is omitted. ∎

The following lemma states that checking a path formula $\varphi(x)$ at a node k_0 belonging to the tree sequence ts corresponds to verifying a number of sentences that do not refer to properties of x relative to the whole tree sequence anymore, but only relative to certain disjoint components of it. In particular, these disjoint substructures are the (above defined) sequences π_1 and π_3, with the trees rooted at them, the path π_2, with the trees rooted at successor nodes that are not on π_2, the node k_0, and the k trees rooted at the k successors of k_0.

LEMMA 5.13
(First Decomposition Lemma)

For every $\mathrm{MPL}_\Sigma[\tau_1]$- formula $\varphi(x)$ of quantifier depth n, there exists a finite set Φ of elements of the form $(\psi_1, \psi_2, a, \beta_0, \ldots, \beta_{k-1}, \psi_3)$, where ψ_i, with $i = 1, 2$, are n-type descriptors in $\mathrm{MFO}_{\Sigma_i}[\tau_4]$, ψ_3 is an n-type descriptor in $\mathrm{MFO}_{\Sigma_1}[\tau_3]$, $a \in \Sigma$, and β_i, with $i = 0, \ldots, k-1$, are n-type descriptors in $\mathrm{MPL}_\Sigma[\tau_2]$, such that, for any $ts \in TS_k(\Sigma)$ and any element k_0 in ts, it holds that:

> $(ts, k_0) \models \varphi(x)$ if and only if there exists $(\psi_1, \psi_2, a, \beta_0, \ldots, \beta_{k-1}, \psi_3)$ in Φ such that $v_i(ts, k_0) \models \psi_i$, for $i = 1, 2, 3$, $ts(k_0) = a$, and $ts_{k_0,i} \models \beta_i$, for $i = 0, \ldots, k-1$.

PROOF. By Corollary 5.11, $\varphi(x)$ is equivalent to a finite disjunction $\bigvee\{\psi^n_{ts,k_0} \mid ts \models \varphi[k_0]\}$. Hence, $(ts, k_0) \models \varphi(x)$ if and only if there exist ts', k_0' such that $(ts', k_0') \models \varphi(x)$ and $(ts, k_0) \models \psi^n_{ts',k_0'}(x)$. By Theorem 5.10, this holds true if and only if there exist ts', k_0' such that $(ts', k_0') \models \varphi(x)$ and $(ts, k_0) \sim_n (ts', k_0')$. This is equivalent to the existence of ts'', k_0'' such that $(ts'', k_0'') \models \varphi(x)$, $v_i(ts, k_0) \sim_n v_i(ts'', k_0'')$, with $i = 1, 2, 3$, $ts(k_0) = ts''(k_0'')$ and $ts_{k_0,i} \sim_n ts''_{k_0'',i}$, with $i = 0, \ldots, k-1$. The implication from right to left follows from Lemma 5.12 by setting $ts' = ts''$ and $k_0' = k_0''$. To prove the opposite implication, take $ts'' = ts$ and $k_0'' = k_0$. We must show that $(ts, k_0) \models \varphi(x)$. Observe that, by hypothesis, $(ts, k_0) \sim_n (ts', k_0')$ and $(ts', k_0') \models \varphi(x)$. Since $\varphi(x)$ is of quantifier depth n, by applying Theorem 5.10, we have that $(ts, k_0) \models \varphi(x)$.

Now, we proceed by invoking the analogous of Theorem 5.10 for words and k-ary trees, to obtain, respectively, appropriate n-type descriptors $\psi_i = \psi^n_{v_i(ts'',k_0'')}$, with $i = 1, 2, 3$, and $\beta_i = \psi^n_{ts''_{k_0'',i}}$, with $i = 0, \ldots, k-1$, such that $v_i(ts, k_0) \models \psi_i$, for $i = 1, 2, 3$, and $ts_{k_0,i} \models \beta_i$, for $i = 0 \ldots k-1$. By collecting all such n-type

descriptors, we obtain a set Φ as required. Since, for every $n \geq 0$, the equivalence relation $\equiv_n$ has finite index, by virtue of Theorem 5.10, there is a finite number of non equivalent n-type descriptors. From this, it follows that the set Φ is finite. ∎

It is possible to prove a similar decomposition lemma for the second-order case. To state it, we need the following definition. Let $ts \in TS_k(\Sigma)$ and P_0 be a full path in ts. We denote by $v(ts, P_0)$ the infinite word over Σ_2 whose i-th letter is $(a, (i_1, j_1), \ldots (i_{k-1}, j_{k-1}))$ if $P_0(i)$ carries letter $a \in \Sigma$ and, for $r = 1, \ldots, k-1$, the tree rooted in the i_r-th son of it has n-type j_r. The proof of Lemma 5.14 is similar to that of Lemma 5.13, and thus omitted.

LEMMA 5.14
(Second Decomposition Lemma)

For every $\mathrm{MPL}_\Sigma[\tau_1]$-formula $\varphi(X)$ of quantifier depth n, there exists a finite set Φ of elements of the form (ψ_1, ψ_2, ψ_3), where ψ_1 is an n-type descriptor in $\mathrm{MFO}_{\Sigma_1}[\tau_4]$, ψ_2 is an n-type descriptor in $\mathrm{MFO}_{\Sigma_2}[\tau_3]$, and ψ_3 is an n-type descriptor in $\mathrm{MFO}_{\Sigma_1}[\tau_3]$, such that, for any $ts \in TS_k(\Sigma)$ and any full path P_0 in ts, it holds that:

$(ts, P_0) \models \varphi(X)$ if and only if there exists (ψ_1, ψ_2, ψ_3) in Φ such that $v_i(ts, k_0) \models \psi_i$, with $i = 1, 3$, and $v(ts, P_0) \models \psi_2$.

We are now ready to prove the main theorem.

THEOREM 5.15
(Expressive completeness of $\mathrm{CTSL}^*_{\mathrm{k}}$)

For every set $T \subseteq TS_k(\Sigma)$ of infinite sequences of Σ-labeled infinite trees, if T is definable in $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$, then T is definable in $\mathrm{CTSL}^*_{\mathrm{k}}$.

PROOF. We prove that every $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$-sentence corresponds to an equivalent $\mathrm{CTSL}^*_{\mathrm{k}}$-formula. Without loss of generality, we consider $\mathrm{MPL}[\tau_1]$-sentences with set quantification restricted to *full paths* only. Let φ be an $\mathrm{MPL}[\tau_1]$-sentence. We focus on the two relevant cases: $\varphi = \exists x \phi(x)$ and $\varphi = \exists X \phi(X)$.

Let $\varphi = \exists x \phi(x)$. By Lemma 5.13, checking $\phi(x)$ in (ts, k_0) is equivalent to checking certain sentences ψ_1, ψ_2, $\beta_0, \ldots, \beta_{k-1}$, and ψ_3, and a label $a \in \Sigma$, taken from a finite set Φ, in particular substructures of ts. It suffices to consider the case in which $|\Phi| = 1$.

The first-order sentence ψ_1 can be mapped into an equivalent PTL-formula h_1 (cf. [16, 18]). Given the formula h_1, we construct the dual formula $h_1^{-1} \in \mathrm{PPTL}$, that is, a formula such that, for every finite word w of length l, $(w, 0) \models h_1$ if and only if $(w, l-1) \models h_1^{-1}$. The formula h_1^{-1} contains atomic propositions $P_{(a,j)}$, for $(a, j) \in \Sigma_1$, which must be replaced by suitable $\mathrm{CTL}^*_{\mathrm{k}}$-formulas $q_{(a,j)}$. Each formula $q_{(a,j)}$ states that a given node satisfies P_a and the tree rooted at it has n-type j, i.e., it satisfies the j-th n-type descriptor. Hereinafter, we will denote by p_j the $\mathrm{CTL}^*_{\mathrm{k}}$-formula equivalent to the j-th n-type descriptor, whose existence is guaranteed by

Theorem 5.3. Let $q_{(a,j)} = P_a \wedge p_j$ and let $(h_1^{-1})'$ be the $\mathrm{PCTSL}^*_{\mathrm{k}}$-formula obtained from h_1^{-1} by replacing propositions $P_{(a,j)}$ by formulas $q_{(a,j)}$ (and using the symbols $\bigcirc$, $\triangle$, $\bigcirc^{-1}$, and $\triangle^{-1}$ for the linear time operators that occur in h_1^{-1}).

In a similar way, the first-order sentence ψ_3 can be mapped into a PTL-formula h_3. The formula h_3 can be turned into a $\mathrm{CTSL}^*_{\mathrm{k}}$-formula $(h_3)'$ by replacing propositions $P_{(a,j)}$ by $\mathrm{CTL}^*_{\mathrm{k}}$-formulas $q_{(a,j)}$ as in the previous case (notice that, in this case, linear past operators are not needed).

Finally, the first-order sentence ψ_2 can be mapped into an equivalent PTL-formula h_2, whose dual version h_2^{-1} is obtained as already explained in the case of h_1. The $\mathrm{PCTL}^*_{\mathrm{k}}$-formula $(h_2^{-1})'$ is obtained by replacing propositions $P_{(a,(i_1,j_1),\ldots,(i_{k-1},j_{k-1}))}$ by formulas $q_{(a,(i_1,j_1),\ldots,(i_{k-1},j_{k-1}))}$ which states that a given node satisfies P_a and the tree rooted at its i_r-th son has n-type j_r, that is, it satisfies the j_r-th n-type descriptor p_{j_r}, for $r = 1, \ldots k-1$. Formally,

$$q_{(a,(i_1,j_1),\ldots,(i_{k-1},j_{k-1}))} = P_a \wedge \bigwedge_{r=1}^{k-1} \mathbf{N_{i_r}} p_{j_r}.$$

As for the $\mathrm{MPL}[\tau_2]$-sentences β_i, with $i = 0, \ldots, k-1$, let b_i be the index of the n-type descriptor β_i. By exploiting once more Theorem 5.3, we obtain a $\mathrm{CTL}^*_{\mathrm{k}}$-formula p_{b_i}, for each $i = 0, \ldots, k-1$.

Merging together the above results, we have that the given $\mathrm{MPL}[\tau_1]$-formula φ is equivalent to the $\mathrm{PCTSL}^*_{\mathrm{k}}$-formula:

$$p_\varphi = \Diamond(\bigcirc^{-1}(h_1^{-1})' \wedge \mathbf{E}\mathbf{F}p \wedge \bigcirc(h_3)'),$$

where

$$p = \mathbf{X}^{-1}(h_2^{-1})' \wedge P_a \wedge \bigwedge_{i=0}^{k-1} \mathbf{N_i} p_{b_i}.$$

Theorem 5.8 guarantees that there exists a $\mathrm{CTSL}^*_{\mathrm{k}}$-formula p'_φ (that is, a formula devoid of past operators) which is equivalent to p_φ, and, thus, equivalent to φ.

Let $\varphi = \exists X \phi(X)$, with quantifier depth of ϕ equal to $n \geq 1$. By Lemma 5.14, checking $\phi(X)$ in (ts, P_0) is equivalent to checking certain sentences ψ_1, ψ_2, and ψ_3 in particular substructures of ts. In analogy to the case of first-order quantification, ψ_1, ψ_2, and ψ_3 can be replaced by a $\mathrm{PCTSL}^*_{\mathrm{k}}$ formula $(h_1^{-1})'$, a $\mathrm{CTL}^*_{\mathrm{k}}$ formula $(h_2)'$, and a $\mathrm{CTSL}^*_{\mathrm{k}}$ formula $(h_3)'$, respectively. It is easy to check that the $\mathrm{MPL}[\tau_1]$-formula φ is equivalent to the $\mathrm{PCTSL}^*_{\mathrm{k}}$-formula:

$$p_\varphi = \Diamond(\bigcirc^{-1}(h_1^{-1})' \wedge \mathbf{E}(h_2)' \wedge \bigcirc(h_3)').$$

Again, by Theorem 5.8, there exists a $\mathrm{CTSL}^*_{\mathrm{k}}$-formula p'_φ which is equivalent to p_φ, and, thus, equivalent to φ. ∎

Putting together Theorems 5.7 and 5.15, we have that $\mathrm{CTSL}^*_{\mathrm{k}}$ is as expressive as $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$, when interpreted over infinite sequences of infinite k-ary trees. In particular, it holds that every $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$-sentence can be encoded into a (sometimes much longer) $\mathrm{CTSL}^*_{\mathrm{k}}$-formula.

5.3 Temporal operators for time granularity

In this section, we show how the basic temporal operators of (horizontal) contextualization, local displacement, and projection, as well as the derived operators of abstraction and vertical contextualization, can be encoded in $\mathrm{CTSL}^*_{\mathrm{k}}$.

We proceed as follows: first, we provide a natural language specification of a property based on one of these temporal operators; then, we encode it in $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$; finally, we express it in $\mathrm{CTSL}^*_{\mathrm{k}}$.

- **Projection**: "whenever p holds at a given time point, then q holds at its ith projection (if any)":
  – $\forall x(\exists y(\uparrow(y) = x) \wedge p(x) \rightarrow \exists y(\downarrow_i(x) = y \wedge q(y)))$;
  – $\Box\mathbf{AG}(\mathbf{X}\top \wedge p \rightarrow \mathbf{N}_i q)$.
  Notice that the formula $\exists y(\uparrow(y) = x)$ is always true in downward unbounded layered structures; however, it can be false in n-layered ones (cf. Section 6).
- **Abstraction**: "whenever p holds at a given time point, then q holds at its abstraction (if any)":
  – $\forall x(\exists y(\uparrow(x) = y) \wedge p(x) \rightarrow \exists y(\uparrow(x) = y \wedge q(y)))$;
  – $\Box\mathbf{AG}(\mathbf{X}p \rightarrow q)$.
- **(Horizontal) contextualization**: "p holds at all time points of the i-th layer":
  – $\forall x(\mathtt{T}^i(x) \rightarrow p(x))$;
  – $\Box\mathbf{A}\mathbf{X}^i p$,
  where $\mathbf{X}^i$ is the concatenation of i next operators $\mathbf{X}$, for every $i \geq 0$.
- **Local successor**: "if p holds at a given time point of the i-th layer, then q holds at its successor":
  – $\forall x(\mathtt{T}^i(x) \wedge p(x) \rightarrow \exists y(y = +_i 1(x) \wedge q(y)))$;
  – for $i = 0$, $\Box(p \rightarrow \bigcirc q)$;
  for $i > 0$, $\Box(\bigwedge_{r=0}^{k^i-2}(\hat{\mathbf{N}}^i_r p \rightarrow \hat{\mathbf{N}}^i_{r+1} q) \wedge (\hat{\mathbf{N}}^i_{k^i-1} p \rightarrow \bigcirc \hat{\mathbf{N}}^i_0 q))$,
  where, for $r, i \geq 0$, $\hat{\mathbf{N}}^i_r = \mathbf{N}_{j_0} \ldots \mathbf{N}_{j_{i-1}}$, with $\sum_{s=0}^{i-1} k^s j_{i-1-s} = r$, that is, $j_0 \ldots j_{i-1}$ is the k-ary representation of r.
- **Local until**: "if h holds at a given time point of the i-th layer, then there exists a time point in its future (belonging to the same layer) at which q holds and p holds until then":
  – $\forall z(\mathtt{T}^i(z) \wedge h(z) \rightarrow \exists y(\mathtt{T}^i(y) \wedge q(y) \wedge \forall x(\mathtt{T}^i(x) \wedge z \leq x < y \rightarrow p(x))))$;
  – $\Box(\bigwedge_{s=0}^{k^i-1}(\hat{\mathbf{N}}^i_s h \rightarrow \bigvee_{r=s}^{k^i-1}(\hat{\mathbf{N}}^i_r q \wedge \bigwedge_{t=s}^{r-1} \hat{\mathbf{N}}^i_t p)) \vee$
  $\bigwedge_{s=0}^{k^i-1}(\hat{\mathbf{N}}^i_s h \rightarrow (\bigwedge_{r=s}^{k^i-1} \hat{\mathbf{N}}^i_r p) \wedge \bigcirc(\mathbf{A}\mathbf{X}^i p \triangle \bigvee_{r=0}^{k^i-1}(\hat{\mathbf{N}}^i_r q \wedge \bigwedge_{t=0}^{r-1} \hat{\mathbf{N}}^i_t p))))$.

- **(Vertical) contextualization**: "p holds at all time points at distance i from the origin of the layer they belong to":
  – $\forall x(\mathtt{D}^i(x) \rightarrow p(x))$;
  – for $i = 0$, $\mathbf{E}(p \wedge \mathbf{G}\mathbf{X_0}p)$;
  – for $i > 0$, $\bigwedge_{j=0}^{\lfloor \log_k(i) \rfloor} \mathbf{X}^{\lfloor i/k^j \rfloor} \hat{\mathbf{N}}^j_{i \mathtt{mod} k^j} p \wedge \mathbf{E}(\mathbf{G}(\mathbf{X_0}\mathtt{true} \wedge \hat{\mathbf{N}}_i^{\lfloor log_k i \rfloor + 1} p))$.

5.4 *Complexity of* $\mathrm{CTSL}^*_{\mathrm{k}}$

We prove now that the satisfiability problem for $\mathrm{CTSL}^*_{\mathrm{k}}$ is 2EXPTIME-complete, and thus elementarily decidable.

THEOREM 5.16
($\mathrm{CTSL}^*_{\mathrm{k}}$ is elementary decidable)

The satisfiability problem for $\mathrm{CTSL}^*_{\mathrm{k}}$ is 2EXPTIME-complete.

PROOF. Hardness follows from 2EXPTIME-hardness of the satisfiability problem for CTL^* [32]. To show that the satisfiability problem for $\mathrm{CTSL}^*_{\mathrm{k}}$ is in 2EXPTIME, we reduce it to the satisfiability problem for $\mathrm{CTL}^*_{\mathrm{k}}$, which has been shown to be 2EXPTIME-complete in Section 5.1 (Theorem 5.6).

To this end, let Σ be a finite alphabet and $\Sigma' = \Sigma \cup \{*\}$, where $*$ is a new symbol not belonging to Σ. We define a translation τ that maps $\mathrm{CTSL}^*_{\mathrm{k}}$-formulas over $\mathcal{P}_\Sigma$ into equisatisfiable $\mathrm{CTL}^*_{\mathrm{k}}$-formulas over $\mathcal{P}_{\Sigma'}$.

For any $\mathrm{CTSL}^*_{\mathrm{k}}$-formula α, let

$$\tau(\alpha) = \mathtt{RightPath}(P_*) \wedge \hat{\tau}(\alpha),$$

where

$$\mathtt{RightPath}(P_*) = P_* \wedge \mathbf{AG}(P_* \rightarrow (\mathbf{N_{k-1}}P_* \wedge \textstyle\bigwedge_{i=0}^{k-2} \mathbf{N_i}\neg P_*) \wedge \neg P_* \rightarrow \mathbf{AX}\neg P_*)$$

and $\hat{\tau}(\alpha)$ is recursively defined as follows:

$$\begin{array}{lcl} \hat{\tau}(\alpha) & = & \mathbf{N_0}\alpha, \text{ whenever } \alpha \text{ monolithic in } \mathrm{CTL}^*_{\mathrm{k}}; \\ \hat{\tau}(\bigcirc \alpha) & = & \mathbf{N_{k-1}}\hat{\tau}(\alpha); \\ \hat{\tau}(\alpha \triangle \beta) & = & \mathbf{E}(\mathbf{G}P_* \wedge \hat{\tau}(\alpha)\mathbf{U}\hat{\tau}(\beta)). \end{array}$$

It is not difficult to show that, for every formula α of $\mathrm{CTSL}^*_{\mathrm{k}}$, α is satisfiable over $TS_k(\Sigma)$ if and only if $\tau(\alpha)$ is satisfiable over $T_k(\Sigma')$.

As for the left to right direction, let α be a formula of $\mathrm{CTSL}^*_{\mathrm{k}}$ and $ts = t_0, t_1, \ldots \in TS_k(\Sigma)$ be a tree sequence that validates α. Consider a tree $t \in T_k(\Sigma')$ built as follows: we label the subtree of t rooted at 0 in the same way in which t_0 is labeled, the subtree rooted at $(k-1)0$ as t_1, the subtree rooted at $(k-1)(k-1)0$ as t_2, and so on. Moreover, we label the set of nodes $\{(k-1)^n \mid n \geq 0\}$ with $* \in \Sigma'$. It is easy to prove, by induction on the structure of α, that t validates $\tau(\alpha)$.

As for the opposite direction, let α be a formula of $\mathrm{CTSL}^*_{\mathrm{k}}$ and $t \in T_k(\Sigma')$ be a tree that validates $\tau(\alpha)$. Let $ts = t_0, t_1, \ldots$ be a tree sequence in $TS_k(\Sigma)$ built as follows: t_0 is labeled as the subtree of t rooted at 0, t_1 is labeled as the subtree of t rooted at $(k-1)0$, t_2 is labeled as the subtree of t rooted at $(k-1)(k-1)0$, and so on. It is easy to prove, by induction on the structure of α, that ts validates α. ∎

6 Coping with n-layered structures

We conclude the paper by showing how to tailor temporal logics for time granularity over downward unbounded layered structures to deal with n-layered structures. n-layered structures are layered structures endowed with a finite and bounded number of temporal domains. Formally, we define a *k-refinable n-layered structure* as a triplet $\langle \bigcup_{i \leq n} T^i, \downarrow, <_{tot} \rangle$, with $n \geq 0$, such that:

- $\{T^i\}_{i \leq n}$ are pairwise disjoint copies of $\mathbb{N}$;
- $\downarrow$: $\{0, \ldots, k-1\} \times \bigcup_{i<n} T^i \to \bigcup_{i \leq n} T^i$ is a bijection such that, for all $i < n$, $\downarrow|_{T^i}$: $\{0, \ldots, k-1\} \times T^i \to T^{i+1}$ is a bijection;
- $<_{tot}$ is such that
  1. $\langle T^0, <_{tot}|_{T^0 \times T^0} \rangle$ is isomorphic to $\mathbb{N}$ with the standard ordering;
  2. $t <_{tot} \downarrow (j, t)$, for $0 \leq j \leq k-1$;
  3. $\downarrow (j, t) <_{tot} \downarrow (j+1, t)$, for $0 \leq j \leq k-2$;
  4. if $t <_{tot} t'$ and $t' \notin \bigcup_{i=0}^{n} \downarrow^i (t)$, then $\downarrow (k-1, t) <_{tot} t'$;
  5. if $t <_{tot} t'$ and $t' <_{tot} t''$, then $t <_{tot} t''$.

  where, $\downarrow^i (t) \subseteq \bigcup_{i \leq n} T^i$ is the set such that $\downarrow^0 (t) = \{t\}$ and, for every $i \geq 1$, $\downarrow^i (t) = \{\downarrow (j, t') : t' \in \downarrow^{i-1} (t), 0 \leq j \leq k-1\}$.

For every $n \geq 0$, n-layered structures can be viewed as infinite sequences of k-ary trees of height n, and vice versa. Moreover, the monadic first-order theory $\mathrm{MFO}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$ can be adopted as the language for time granularity over n-layered structures or, equivalently, over infinite sequences of finite trees of height n. $\mathrm{MFO}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$ is indeed expressive enough to capture the granular primitives of (horizontal) contextualization, local displacement, and projection as well as the derived operators of abstraction and vertical contextualization. Projection, abstraction, (horizontal) contextualization, and local displacement can be expressed as in the case of downward unbounded layered structures. As for vertical contextualization, its definition can be simplified as follows:

$$\mathtt{D}^0(x) = \bigvee_{i=0}^{n} \downarrow_{0^i} (0_0) = x,$$

where 0_0 is the (first-order definable) origin of layer zero.

For all $i > 0$, $\mathtt{D^i}(x)$ can be defined as follows:

$$\mathtt{D^i}(x) = \bigvee_{j=0}^{n} \exists y(\mathtt{T^j}(y) \wedge \mathtt{D^0}(y) \wedge +_{\mathtt{j}}\mathtt{i}(y) = x).$$

REMARK 6.1
Unlike the case of downward unbounded structures (cf. Remark 3.3), by exploiting the finiteness of the layered structure, we can capture the predicate `EqualT` as follows:

$$\mathtt{EqualT}(x, y) = \bigvee_{i=0}^{n} (\mathtt{T^i}(x) \wedge \mathtt{T^i}(y)).$$

On the contrary, there is no way to define the predicate `EqualD`. Indeed, it is possible to show that the addition of such a predicate to $\mathrm{MFO}[<_{tot}, (\downarrow_i)_{i=0}^{k-1}]$ would make the theory undecidable.

Let $\mathrm{CTSL}_{\mathrm{k}}^{\mathrm{n}}$ be the temporal logic for time granularity over infinite sequences of finite trees of height n (n-layered structures). From a syntactical point of view, it coincides with $\mathrm{CTSL}_{\mathrm{k}}^{*}$; however, it is interpreted over infinite sequences of finite k-ary trees of height n (with a strong interpretation for the next operator).

In the following, we show how to reduce the satisfiability problem for $\mathrm{CTSL}_{\mathrm{k}}^{\mathrm{n}}$ to that for $\mathrm{CTSL}_{\mathrm{k}}^{*}$. Let $i \geq 0$ and $\theta_i = \Box(\bigwedge_{j=0}^{i} \mathbf{AX}^j P_* \wedge \mathbf{AX}^{i+1}\mathbf{AG}\neg P_*)$, where $* \notin \Sigma$. The translation τ from $\mathrm{CTSL}_{\mathrm{k}}^{*}$-formulas to $\mathrm{CTSL}_{\mathrm{k}}^{*}$-formulas can be defined as follows:

$$\begin{array}{lcl}
\tau(P_a) & = & P_a \text{ for } a \in \Sigma; \\
\tau(\alpha \wedge \beta) & = & \tau(\alpha) \wedge \tau(\beta); \\
\tau(\neg\alpha) & = & \neg\tau(\alpha); \\
\tau(\bigcirc \alpha) & = & \bigcirc \tau(\alpha); \\
\tau(\alpha \triangle \beta) & = & \tau(\alpha) \triangle \tau(\beta); \\
\tau(\mathbf{A}\alpha) & = & \mathbf{A}\tau(\alpha) \text{ whenever the outermost operator of } \alpha \text{ (if any) belongs to} \\
 & & \{\wedge, \neg, \mathbf{A}, \mathbf{E}\}; \\
\tau(\mathbf{AX}\alpha) & = & \mathbf{AX}(P_* \rightarrow \tau(\alpha)); \\
\tau(\mathbf{AX_i}\alpha) & = & \mathbf{AX_i}(P_* \rightarrow \tau(\alpha)); \\
\tau(\mathbf{A}(\alpha\mathbf{U}\beta)) & = & P_* \rightarrow \mathbf{A}(\tau(\alpha)\mathbf{U}(P_* \wedge \tau(\beta))); \\
\tau(\mathbf{E}\alpha) & = & \mathbf{E}\tau(\alpha) \text{ whenever the outermost operator of } \alpha \text{ (if any) belongs to} \\
 & & \{\wedge, \neg, \mathbf{A}, \mathbf{E}\}; \\
\tau(\mathbf{EX}\alpha) & = & \mathbf{EX}(P_* \wedge \tau(\alpha)); \\
\tau(\mathbf{EX_i}\alpha) & = & \mathbf{EX_i}(P_* \wedge \tau(\alpha)); \\
\tau(\mathbf{E}(\alpha\mathbf{U}\beta)) & = & P_* \wedge \mathbf{E}(\tau(\alpha)\mathbf{U}(P_* \wedge \tau(\beta))).
\end{array}$$

For every $\mathrm{CTSL}_{\mathrm{k}}^{*}$-formula α and $i \geq 0$, let $\eta_i(\alpha) = \theta_i \wedge \tau(\alpha)$. It is not difficult to show that, for every $n \geq 0$, the satisfiability of α over infinite sequences of finite k-ary trees of height n can be reduced to the satisfiability of $\eta_n(\alpha)$ over infinite sequences of infinite k-ary trees. This is formally stated by the following theorem.

THEOREM 6.2
For every $\mathrm{CTSL}^{*}_{\mathrm{k}}$-formula α and $n \geq 0$, there exists an n-layered structure satisfying α if and only if there exists a downward unbounded layered structure satisfying $\eta_n(\alpha)$.

As an immediate corollary of Theorem 6.2, we have that the satisfiability problem for $\mathrm{CTSL}^{\mathrm{n}}_{\mathrm{k}}$ is elementarily decidable. As for the expressiveness of $\mathrm{CTSL}^{\mathrm{n}}_{\mathrm{k}}$, the proof given for $\mathrm{CTSL}^{*}_{\mathrm{k}}$ also works for $\mathrm{CTSL}^{\mathrm{n}}_{\mathrm{k}}$. Hence, $\mathrm{CTSL}^{\mathrm{n}}_{\mathrm{k}}$ is as expressive as $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$ over infinite sequences of finite k-ary trees of height n. However, since second-order quantification in $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$ over sequences of finite trees is restricted to *finite* paths (of bounded length), it is not difficult to see that $\mathrm{MPL}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$ and $\mathrm{MFO}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$ have the same expressive power over infinite sequences of finite k-ary trees of height n. Therefore, $\mathrm{CTSL}^{\mathrm{n}}_{\mathrm{k}}$ is as expressive as $\mathrm{MFO}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$.

7 Conclusions and Future Work

In this paper we proposed a temporal logic for time granularity, that we called $\mathrm{CTSL}^{*}_{\mathrm{k}}$, which is based on a simple combination of the branching time logic $\mathrm{CTL}^{*}_{\mathrm{k}}$ (the extension of CTL^{*} with k directed successors) and the linear time logic PTL. We proved that $\mathrm{CTSL}^{*}_{\mathrm{k}}$ is an expressively complete and elementarily decidable fragment of the theory of k-refinable downward unbounded layered structures with set quantification restricted to infinite paths. Furthermore, according to the adopted combining logic perspective, we have that an axiomatisation, as well as model and satisfiability checking procedures, for $\mathrm{CTSL}^{*}_{\mathrm{k}}$ can be synthesized from the same tools for the component logics. Finally, we showed that (a minor variant of) $\mathrm{CTSL}^{*}_{\mathrm{k}}$ is expressively complete with respect to the first order-theory of k-refinable n-layered structures, and that it is still elementarily decidable. We are currently working at an extension of $\mathrm{CTSL}^{*}_{\mathrm{k}}$ able to capture the full power of the second-order theory of downward unbounded layered (resp. n-layered) structures, preserving its elementary decidability.

References

[1] R. Alur and T.A. Henzinger. Real-time logics: complexity and expressiveness. *Information and Computation*, 104:35–77, 1993.

[2] C. Bettini, S. Jajodia, and X. Wang. *Time Granularities in Databases, Data Mining, and Temporal Reasoning*. Springer, 2000.

[3] P. Blackburn and M. de Rijke. Special issue on combining structures, logics, and theories. *Notre Dame Journal of Formal Logic*, 37, 1996.

[4] J.R. Büchi. On a decision method in restricted second-order arithmetic. In *Proc. 1st International Congress on Logic, Methodology, and Philosophy of Science*, pages 1–11. Stanford Univ. Press, 1962.

[5] S. Buvac and I.A. Mason. Propositional logic of context. In *Proceedings of AAAI*, Washington, 1993. MIT Press.

[6] C.E. Dyreson and R.T. Snodgrass. Temporal granularity. In R. T. Snodgrass, editor, *The TSQL2 Temporal Query Language*, pages 347–385. Kluwer Academic Press, 1995.

[7] E.A. Emerson and C.S. Jutla. The complexity of tree automata and logics of programs. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 328–337. IEEE, 1988.

[8] H.D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 1995.

[9] E. A. Emerson and A. P. Sistla. Deciding full branching time logic. *Information and Control*, 61(3):175–201, 1984.

[10] E.A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B*, pages 995–1072. Elsevier Science Publishers B.V., 1990.

[11] J. Fiadeiro and T. Maibaum. Sometimes "tomorrow" is "sometimes" - action refinement in a temporal logic of objects. In *Proc. of the 1st International Conference on Temporal Logic (ICTL), LNAI 827*, pages 48–66. Springer, 1994.

[12] M. Finger and D. Gabbay. Adding a temporal dimension to a logic system. *Journal of Logic, Language and Information*, 1:203–233, 1992.

[13] M. Finger and D. Gabbay. Combining temporal logic systems. *Notre Dame Journal of Formal Logic*, 37:204–232, 1996.

[14] M. Finger and M. A. Weiss. The unrestricted addition of a temporal dimension to a logic system. In *Proceedings of the 3rd International Conference on Temporal Logic (ICTL)*, pages 53–63, 2000.

[15] M. Franceschet, A. Montanari, and M. de Rijke. Model checking for combined logics. In *Proceedings of the 3rd International Conference on Temporal Logic (ICTL)*, pages 65–73, 2000.

[16] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proc. 7th ACM Symposium on Principles of Programming Languages*, pages 163–173, 1980.

[17] T. Hafer and W. Thomas. Computation tree logic CTL* and path quantifiers in the monadic theory of the binary tree. In Thomas Ottmann, editor, *Automata, Languages and Programming, 14th International Colloquium*, volume 267 of *Lecture Notes in Computer Science*, pages 269–279, Karlsruhe, Germany, 13–17 July 1987. Springer-Verlag.

[18] H. Kamp. *Tense Logic and the Theory of Linear Order*. PhD Dissertation, University of California, Los Angeles, 1968.

[19] H. Läuchli and C. Savoiz. Monadic second-order definable relations on the binary tree. *Journal of Symbolic Logic*, 52:219–226, 1987.

[20] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems (Safety)*. Springer, 1995.

[21] A. Montanari. *Metric and Layered Temporal Logic for Time Granularity*. ILLC Dissertation Series 1996-02, Institute for Logic, Language and Computation, University of Amsterdam, 1996.

[22] A. Montanari and M. de Rijke. Two-sorted metric temporal logic. *Theoretical Computer Science*, 183:187–214, 1997.

[23] A. Montanari, A. Peron, and A. Policriti. Decidable theories of ω-layered metric temporal structures. *Logic Journal of the IGPL*, 7(1):79–102, 1999.

[24] A. Montanari, A. Peron, and A. Policriti. Extending Kamp theorem with binary operators to model time granularity. In *Proceedings of the 3rd International Conference on Temporal Logic (ICTL)*, pages 135–144, 2000.

[25] A. Montanari, A. Peron, and A. Policriti. The taming (timing) of the states. *Logic Journal of the IGPL*, 8(5):681–699, 2000.

[26] A. Montanari and A. Policriti. Decidability results for metric and layered temporal logics. *Notre Dame Journal of Formal Logic*, 37:260–282, 1996.

[27] N. Rescher and J. Garson. Topological logic. *Journal of Symbolic Logic*, 33:537–548, 1968.

[28] S. Safra. On the complexity of ω-automata. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 319–327, White Plains, New York, 24–26 October 1988. IEEE.

[29] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

[30] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B*, pages 133–191. Elsevier Science Publishers, 1990.

[31] W. Thomas. Languages, automata and logic. In G. Rozenberg and Salomaa A., editors, *Handbook of formal languages, Vol. III*, pages 389–455. Springer, 1997.

[32] M. Y. Vardi and L. Stockmeyer. Improved upper and lower bounds for modal logics of programs. In *ACM Symposium on Theory of Computing (STOC)*, pages 240–251, Baltimore, USA, May 1985. ACM Press.

Received 23 December 2000.